The MagPi

Issue 121 | September 2022 | magpi.cc

The official Raspberry Pi magazine

LEARN ELECTRONICS WITH PICO W

YOUR DEFINITIVE GUIDE TO EXPERIMENTING WITH ELECTRONICS

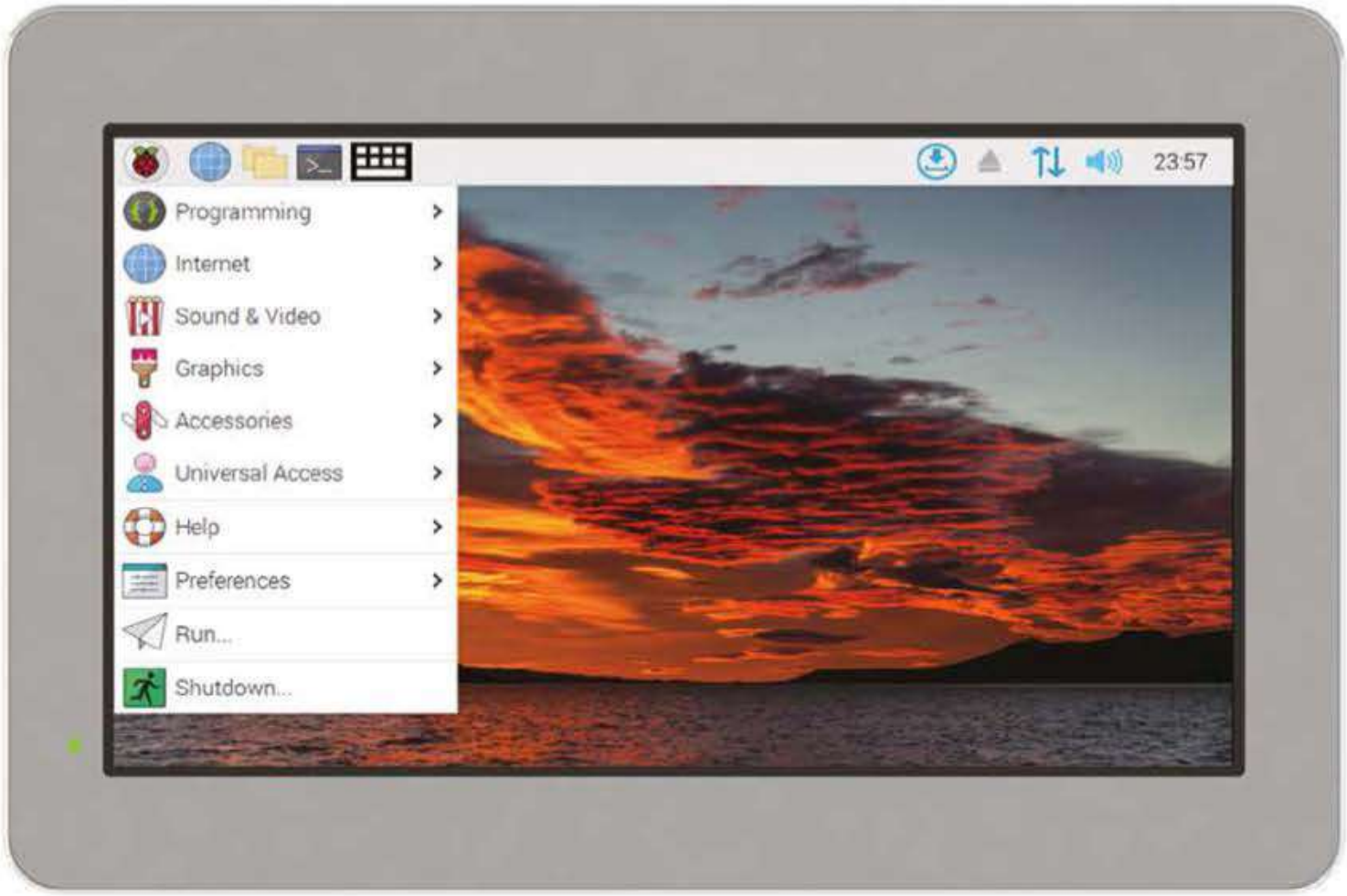
BUILD A **RASPBERRY PI RADIO**

CODE YOUR OWN **VIRTUAL PET**

HAUNTED HALLOWEEN PARTY

54 PAGES OF PROJECTS & TUTORIALS

Industrial Raspberry Pi



ComfilePi

The ComfilePi is a touch panel PC designed with high-tolerant components and no moving parts for industrial applications. It features a water-resistant front panel, touchscreen, color LCD (available in various sizes), RS-232, RS-485, Ethernet, USB, I2C, SPI, digital IO, battery-backed RTC (real-time clock), and piezo buzzer.

Use the rear-panel 40-pin GPIO header to expand its features and capabilities with additional I/O boards. The ComfilePi is UL Listed and employs Raspberry Pi Compute Module.

POWERED BY Raspberry Pi

Visit www.comfiletech.com

COMFILE TECHNOLOGY

WELCOME
to *The MagPi* 121

Electricity is fantastic stuff! It's used to power labour-saving gizmos, light up our streets, move our cars, and perform everyday miracles that would seem the stuff of magic to our ancestors.

Understanding electronics is half of the code and digital-making ethos that underpins Raspberry Pi. And few devices are better for learning electronics than Pico W. Designed to sit right at the heart of electronic projects, Pico is a low-power microcontroller development board with 40 connections on the side – you can connect Pico to just about anything.

Learning electronics is a rite of passage for new Raspberry Pi aficionados. Start by lighting up lights, then create code that responds to buttons, wire up sensors, pull in information from the internet, and build wild creations (**page 30**).

Getting an understanding of code and electronics is an incredibly important skill to have. With it, you can control, and understand, all the incredible things around you that make the modern world work.

Lucy Hattersley Editor

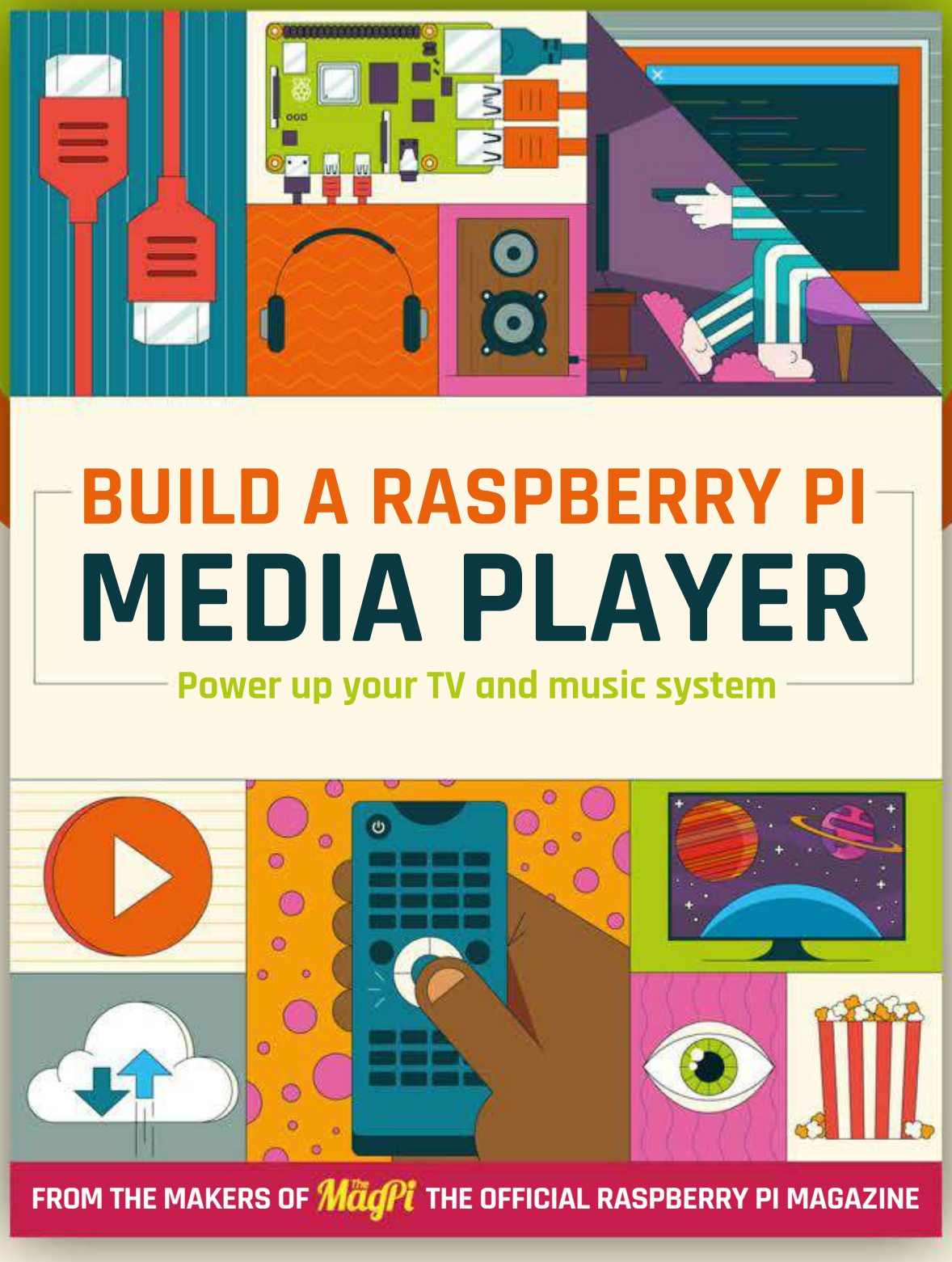
EDITOR

Lucy Hattersley

Lucy is editor of *The MagPi* magazine and is currently sitting in the Greenwich Maritime Museum wondering how they made do with sails. There are some seriously impressive figureheads on the wall beside her though.

@LucyHattersley

Your FREE guide to making a smart TV



magpi.cc/mediaplayer

Contents

Issue 121 • September 2022

Cover Feature

Regulars

Project Showcases

Fireballs Aotearoa

Boost-Box 0.1

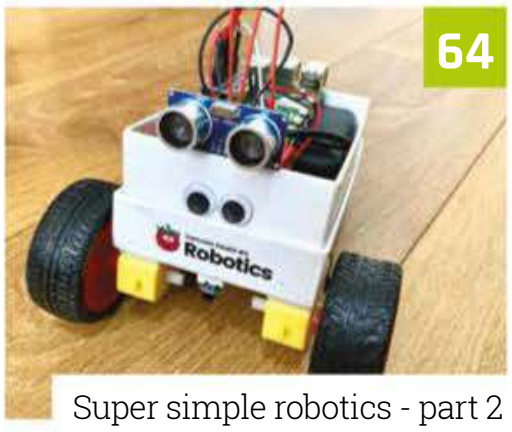
The MagPi magazine is published monthly by Raspberry Pi Ltd. Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701, is the mailing agent for copies distributed in the US. Application to mail at Periodicals prices is pending at Williamsport, PA. POSTMASTER: Send address changes to The MagPi magazine c/o Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701.

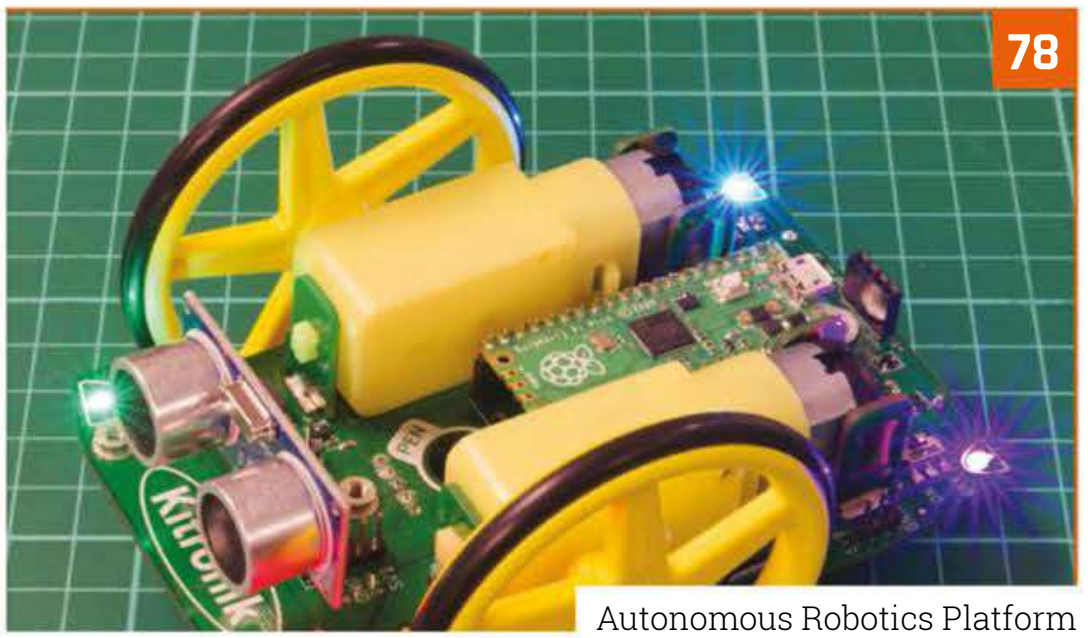
Tutorials

The Big Feature

Reviews

Community





WIN A CROWPI L **BASIC KIT**

NEW PRODUCTS

START HERE

Looking for the latest products? Look no further. With over 400,000 new products stocked, we've got your electronic component and automation needs covered.

Find it at **digikey.co.uk/new** or call 0800 587 0991.

Digi-Key ELECTRONICS

ZeroBug

An inexpensive 3D-printable hexapod that can be programmed or directly controlled. **Rob Zwetsloot** checks its micro servos

MAKER

Maximilian Kern

A hardware developer based in Germany, who likes to do DIY projects in his spare time, including hardware, software and mechanical design.

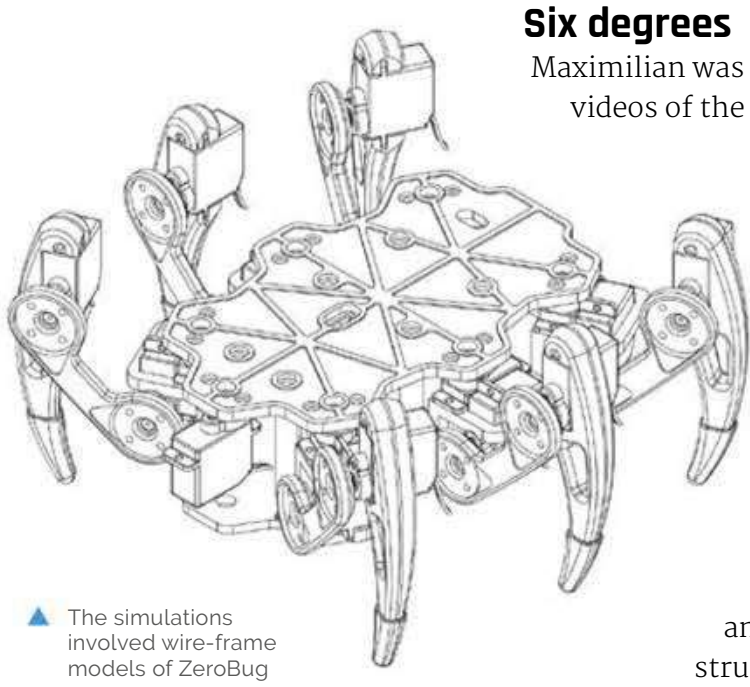
magpi.cc/zerobug

In issue 119 (magpi.cc/119), we reviewed the intimidating SpiderPi, a big hexapod robot that tormented our editor's poor cat. Shortly after the release of the issue, Maximilian Kern emailed us about the one he'd created himself, called ZeroBug, appropriately powered by a Raspberry Pi Zero.

"Zero is in charge of the input methods," Maximilian explains. "It runs a custom web interface and an instance of Pygame. This makes it possible to control the robot using a mouse, keyboard, multitouch, or simply an Xbox gamepad… Instead of specialised robotics servos, this robot uses inexpensive micro servos. Inside its 3D-printed frame there is just enough room for Raspberry Pi, together with a custom PCB for the microcontroller and servo driver."

The microcontroller is an STM32, which is an ARM-based system which controls the leg locomotion through 18 of the affordable servos.

"All of these calculations run at 50Hz, enabling the hexapod to move smoothly and with high precision."



▲ The simulations involved wire-frame models of ZeroBug

Six degrees

Maximilian was inspired by earlier videos of the Boston Dynamics robot experiments, and loved the idea of making robots with legs – starting with a quadrupedal creation of his own, before moving to six legs.

"It turned out to be really difficult to develop proper walking gaits, and the servos seemed to struggle with the weight of the robot," he tells us. "The decision to go with four legs instead of six was mainly due to the cost of servo motors. However, six-legged robots have a big advantage: unlike quadrupeds, they can lift three of their legs while the remaining legs form a stable tripod. This eliminates the need for constant weight shifting and balancing."

The decision to use Raspberry Pi was due to how easily you can connect Bluetooth controllers to it, making it more accessible than RC controllers. With this in mind, Maximilian started simulating his robot.

"With the simulation done, I went on to build the physical hexapod robot," he says. "Since there are 18 servos needed for this hexapod, they define the total cost of the robot. I settled for some cheap Emax ES08A II micro servos which are quite powerful for their size. I only paid around €80 for the entire set of servos. When using proper smart servos for robotics, a single unit can cost this much."

To drive the servos, 18 PWM outputs are needed. "I decided to use the STM32F103 as a microcontroller as it is Arduino-compatible, and I had already gathered some experience with my macro keyboard (**magpi.cc/macrokeypad**). To connect the microcontroller, PWM driver, and Raspberry Pi, I designed a custom PCB that plugs into the back of the GPIO header on Raspberry Pi. To save space, the connector only uses GPIO 1 though 10, which conveniently include 5V, 3.3V, ground, UART, and a couple extra I/O. Voltage regulators on the custom PCB enable the microcontroller and Raspberry Pi to be powered from the battery pack. Both Zero W and the custom board are mounted between the servos, so that the USB port can be accessed from the outside."

Scuttling along

Maximilian claims that building a walking robot is not that hard; instead, making it look right while walking can be a challenge.

Getting the right walk cycle is tricky – it can easily look unnatural

NiMH batteries are used over lithium alternatives as they offer more power

With the camera and mandible, it can pick stuff up and move it around

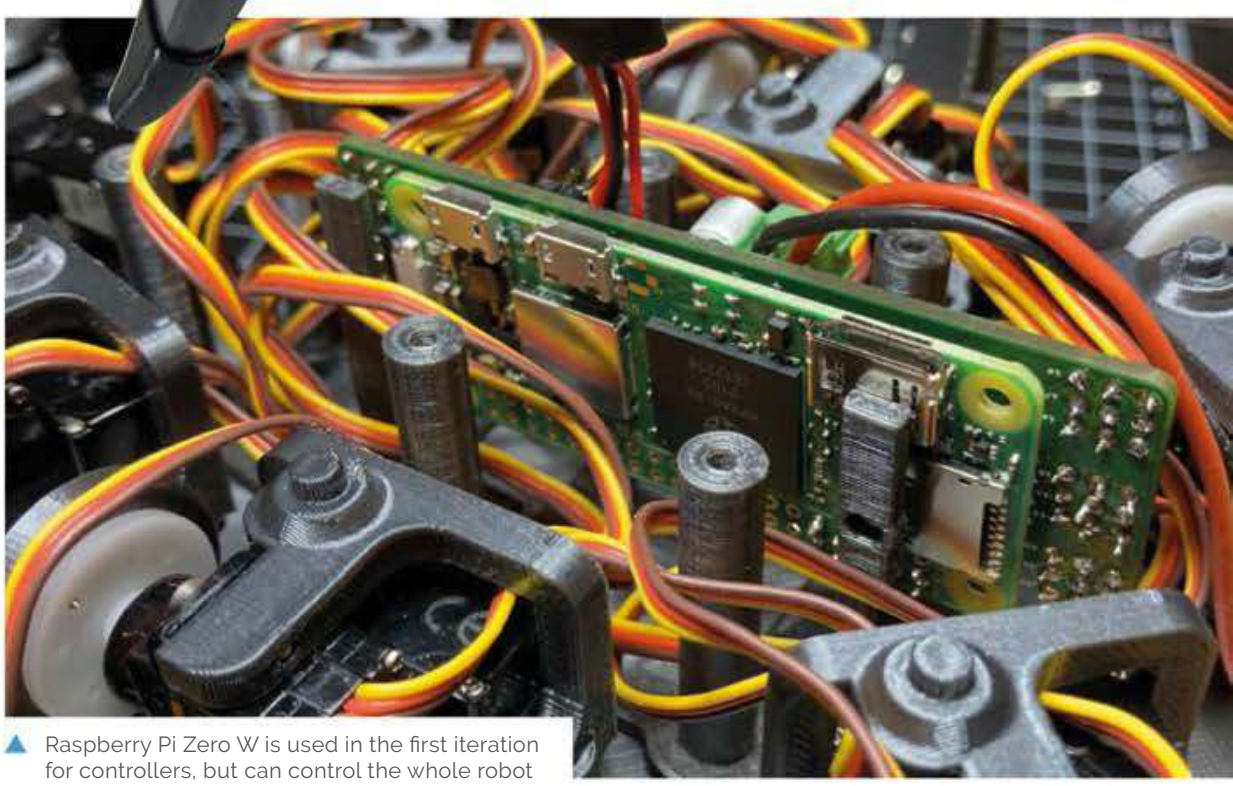
Quick **FACTS**

- There's already a Raspberry Pi-only version of the robot
- The simulator was built in Processing...
- ...and was used to add complexity until it moved properly
- It's also inspired by Matt Denton's uBug (**magpi.cc/ubug**)...
- ... and even the custom 3D-printed parts are based on that design

> Six-legged robots have a big advantage: unlike quadrupeds they can lift three of their legs while the remaining legs form a stable tripod

"Overall, I am really happy with how this project turned out," Maximilian tells us. "I actually started working on a simulator in 2014 and shelved the whole project out of frustration, only to dig it up a few years later. Just at the start of last year it really clicked, and I got the motivation to go through with it."

You can read a lot more about his development process on his Hackaday page (**magpi.cc/zerobug**), and he also has some ideas on how to improve it in the future.



Raspberry Pi Zero W is used in the first iteration for controllers, but can control the whole robot

Big Mouth Billy Bass

An attention-seeking interactive fish gets a Pico W update and becomes an online star, discovers **Rosie Hattersley**

MAKER

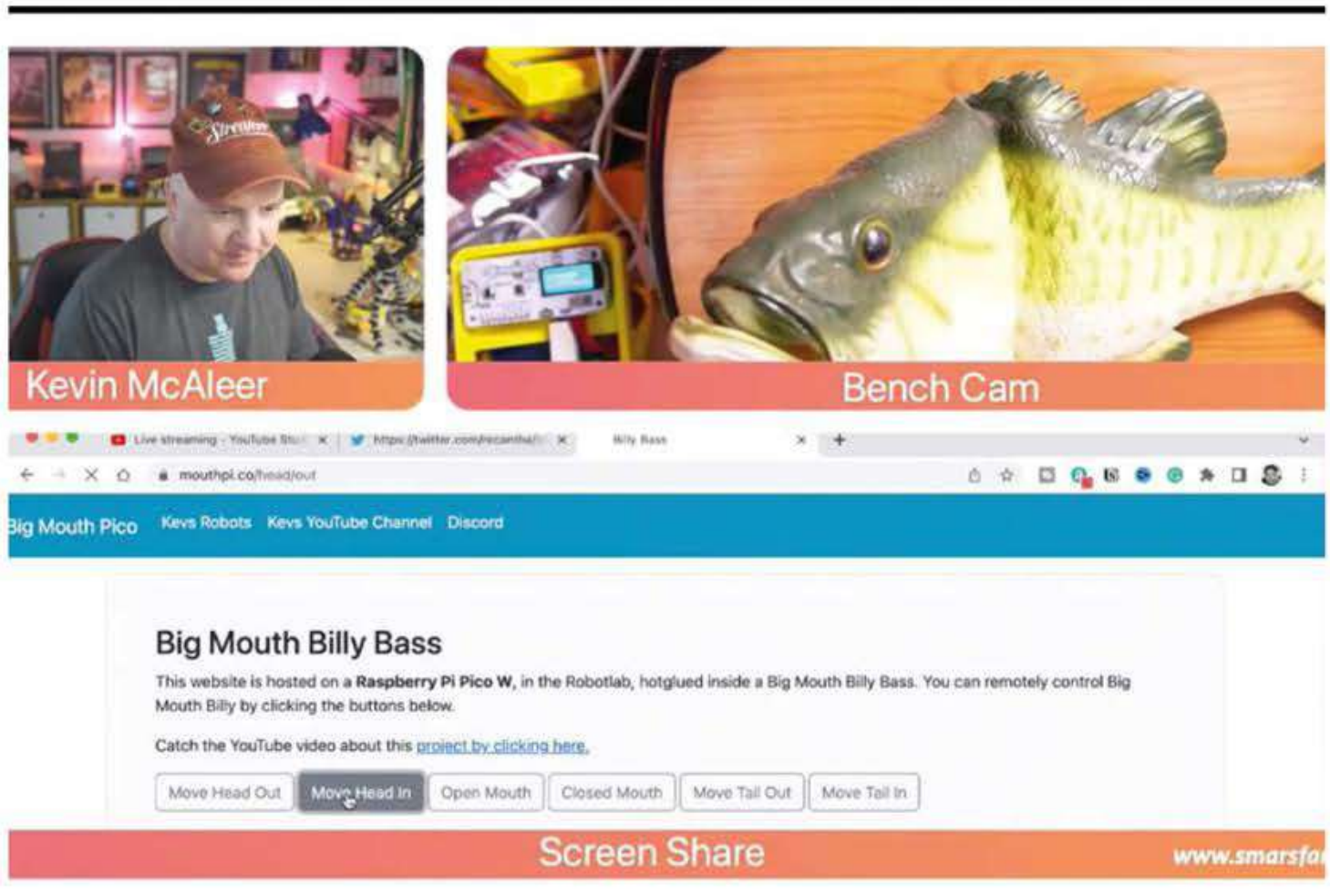
Kevin McAleer

Kevin McAleer makes robots, brings them to life with code, and makes videos about them on YouTube.

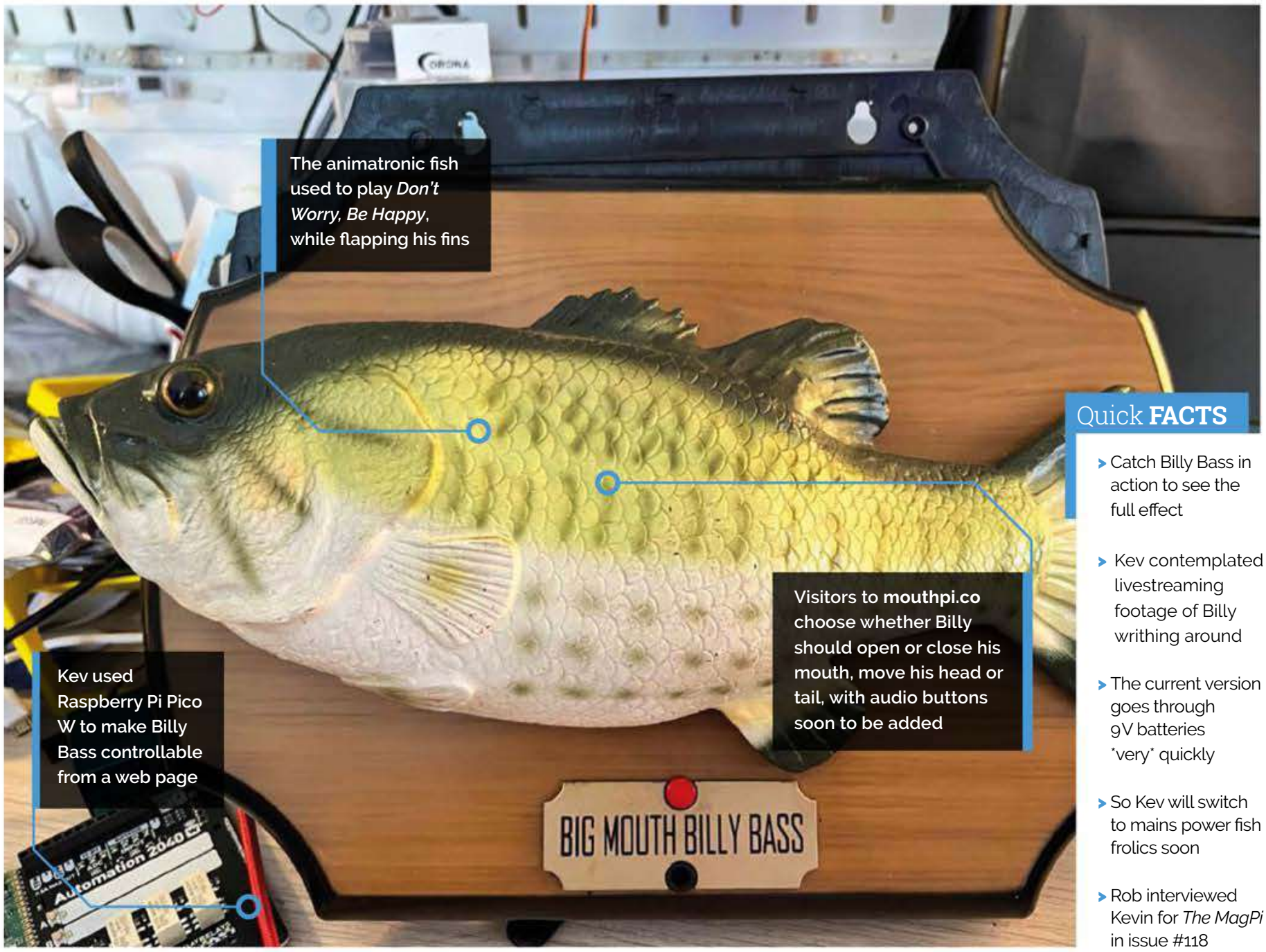
magpi.cc/kevinmcaleer

In the early 2000s, Big Mouth Billy Bass – a kitschy, 3D plastic fish mounted in a picture frame that appears to sing, as well as writhe around – became such an in-demand item that there's reputedly one hanging proudly above the grand piano at the Queen's Balmoral residence. YouTuber, and Raspberry Pi enthusiast, Kevin McAleer relates this apocryphal tale while introducing his latest Raspberry Pi project: a Pico W-controlled update to the tuneful animatronic sea creature, hosted at **mouthpi.co**.

Kevin first fell in love with computing when he became the proud owner of a ZX Spectrum back in 1982. He went on to study computer science. He had a similarly Damascene encounter when he got his first Raspberry Pi not long after it first launched. "Raspberry Pi has helped me learn and master Linux and inspired me to learn Python,



The site being constructed during Kevin's weekly YouTube broadcast



Quick **FACTS**

- Catch Billy Bass in action to see the full effect
- Kev contemplated livestreaming footage of Billy writhing around
- The current version goes through 9V batteries *very* quickly
- So Kev will switch to mains power fish frolics soon
- Rob interviewed Kevin for *The MagPi* in issue #118

which is now my go-to language for all projects." Every Sunday, Kevin hosts a YouTube series (head to **magpi.cc/kevinmcaleer**) discussing all things Raspberry Pi, and is also an accomplished robot builder.

> "Raspberry Pi has helped me learn and master Linux"

Big Mouth strikes again

When Pico W launched in June, Kevin was keen to put the wireless-enabled microcontroller through its paces. Several Pico W web-page-control projects appeared online, but Kevin felt they didn't show the new product's full abilities. He'd previously bought a Big Mouth Billy Bass from eBay for around £20, and reasoned pairing it with Pico W might help him "stretch its capabilities beyond common expectations."

▲ Kevin wanted to show that a Pico W-powered web page could be more than plain text

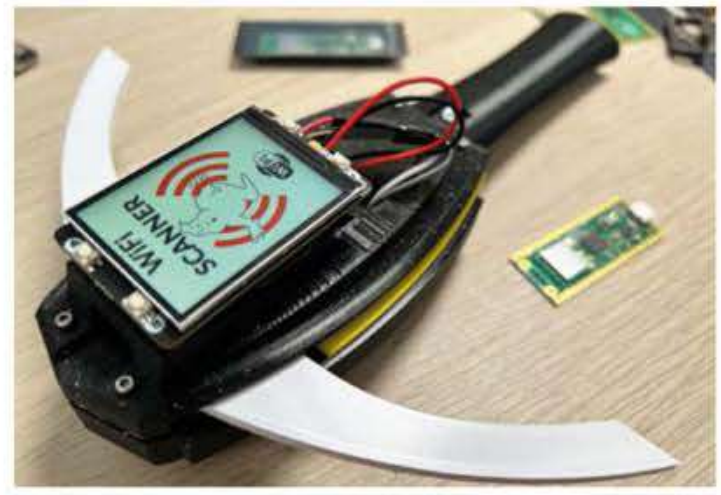
As a child of the 1980s, Kevin couldn't resist designing a Raspberry Pi-based *Ghostbusters* wireless scanner

He wanted to see how well it would hold up with thousands of web page requests per day, and to see how well the web pages could handle colour, fonts, and style sheets. "Pico W doesn't have an OS and has minimal memory, so being able to host a website and control a robot simultaneously is quite remarkable," says Kevin. The **mouthpi.co** site is hosted on the Pico W, which is in turn hidden within the fish robot's body.

Kevin contemplated livestreaming footage of Billy writhing around, but the current iteration of the site has buttons that the user can press to initiate preset movements relating to the head, tail, and mouth. A replacement for the audio files containing the original Big Mouth Billy Bass theme tunes – *Don't Worry Be Happy* and *Take Me To The River* – is planned for the next version. Kevin has also promised his YouTube followers a Furby-based Pico W-controlled site.

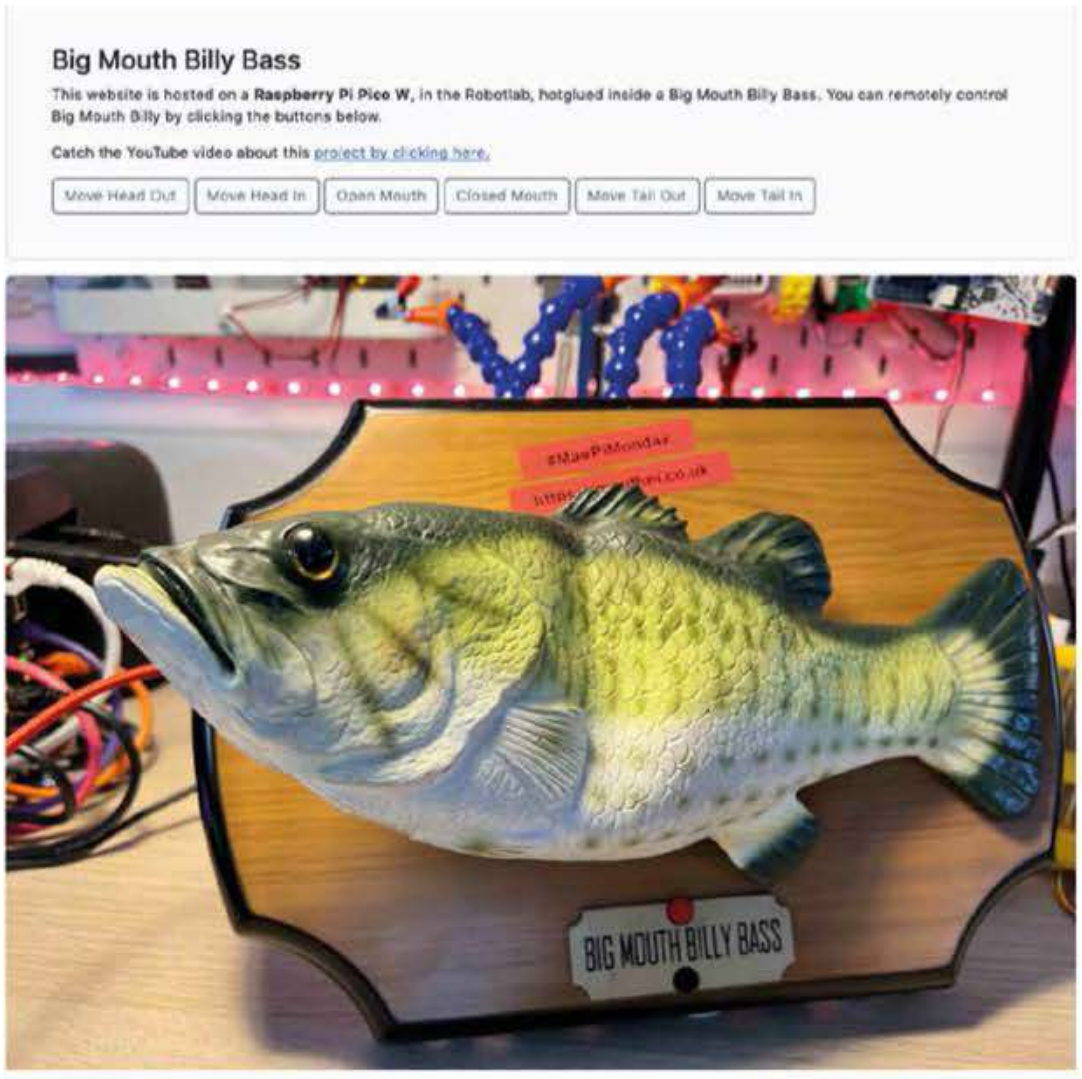
Hacking the hardware

One of the key aspects of this project was establishing how the existing animatronic fish worked. Online research revealed some details, usually with a view to controlling the fish with Alexa, whereas Kevin's plan was to control the motors himself. However, a tear-down of Billy Bass's components, in which Kevin stripped out the existing wiring, showed a relatively simple circuit with three motors.



The WiFi scanner spreads its arms to show signal strength

Having learnt these were "cheap 5 V DC motors", Kevin was confident he'd be able to drive them with a couple of L298N H-bridge modules. He secured them along with Pico W on a mounting



▲ When website visitors click a button to change Billy's pose, the relevant photo is shown

plate to hold them in place. These would allow him to control powerful motors simply by making a GPIO pin on the Pico high or low (1 or 0). Code shared by Raspberry Pi's Alasdair Allan for making an LED light up came in useful here, as did the

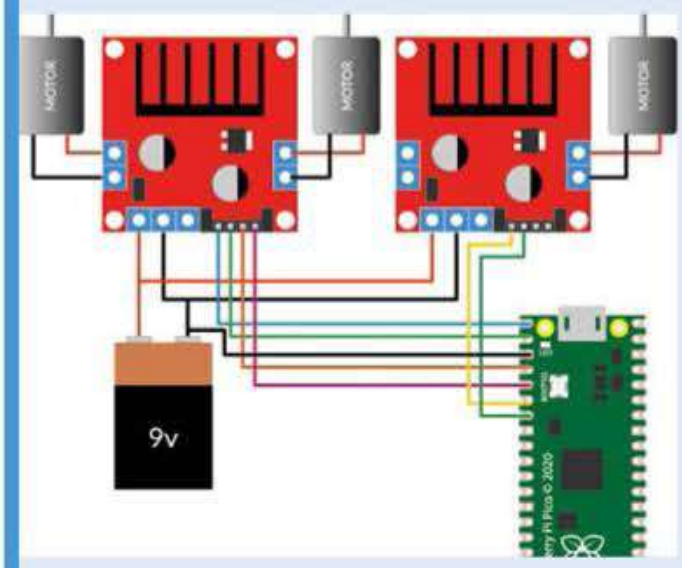
> Kevin was confident he'd be able to drive them with a couple of L298N H-bridge modules

realisation that, as well as sharing the 9 V battery between the motors, the battery ground needed to be connected to the Pico W too.

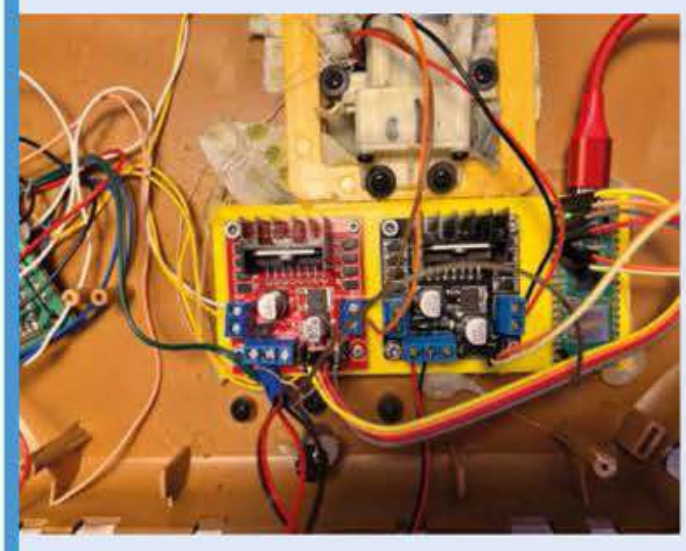
The entire setup cost approximately £20, with a further £20 for the domain name and Cloudflare-hosted website (which offers DDoS protection) covering the next five years. Full MicroPython code and setup instructions are at **magpi.cc/bigmouthwifi**.

Meanwhile, Kevin is already well on his way to his next Pico W project: a *Ghostbusters* PKE WiFi scanner that moves its arms to indicate the strength of the available wireless connection.

In a flap



01 Source a Billy Bass or a similar robot. Disassemble it to see what's inside and sketch out the proposed schematic. Here, H-bridges are used to drive the robot's three motors.



02 Kevin used Pico W and MicroPython to set up a 'billy' class, so moving the head and tail and opening and closing the mouth requires simple commands such as 'billy.open_mouth' or 'billy.flap_tail(3)' to flap the tail three times.

03 Prior to reassembling the Billy Bass case, Kevin used a Pimoroni Pico Explorer to test the wiring. The top white motor moved the head, as expected, while the others made the tail flap.

Digital Zoetrope

Brian Corteil has created a fresh spin on an age-old idea and it's made **David Crookes** dizzy with excitement

MAKER

Brian Corteil

Brian is a maker and amateur robot designer with a "toy age" of eight. He's a member of Cambridge Makespace.

@CannonFodder

Zoetropes were invented by a mathematician called William Horner in 1834 and they proved to be an effective way to produce animations. A set of still images would line the inside of a rotating drum, and viewers would peer through narrow slits in the side. The speed at which they passed the eyes would produce an illusion of movement. As the years went by, the technique was refined and zoetropes remain as much fun today as they did nearly 190 years ago.

As if to prove the concept still has legs, Brian Corteil has created his own version. "I've found zoetropes and optical illusions fascinating since I was a child," he tells us, having worked on his latest one for a recent EMF Camp in the hope of inspiring children at STEM/STEAM events. "They're perfect for showing how persistent vision works and I love playing with old and new technology, combining them into an art project."

Spinning around

Brian's previous attempt made use of twelve OLED displays, each displaying a different still from Eadweard Muybridge's photographic studies of a galloping horse taken in 1878. "I've used his images on several occasions for my projects," he says. "If you reduce the number of pixels, even to a 16×16 LED matrix, you can still see the details of a rider on a horse. They're ideally suited for displaying on a zoetrope."

His first digital zoetrope was not interactive, however, and it needed to be spun by hand. "The OLED displays also required a custom PCB designed to be linked together as a distributed shift register," he says. But although the displays could be updated via a connected Raspberry Pi computer, the results weren't perfect. "When the zoetrope was spun, a dark diagonal line was visible as the OLED displays were being refreshed."

The answer, he surmised, was e-ink displays. "They overcame the issue by not needing to be refreshed to continue displaying a static image," he adds. "I also liked how e-ink displays mirrored the look of paper." For these, Brian used 15 Pimoroni Badger 2040s – fast updating, programmable badges with e-ink displays that utilise the RP2040 microcontroller. He also used a Raspberry Pi Pico board to control the motor that spins the zoetrope. It monitors the safety emergency stop buttons too.

The zoetrope has a slip-ring in the middle of the Lazy Susan bearing. It maintains a power connection when spinning by using a brush or wiper in contact with a rotating ring

Getting animated

Brian designed the zoetrope using the CAD program SolidWorks, creating outlines to be laser-cut from 3 mm and 5 mm plywood. Parts that couldn't be made this way were 3D-printed, and the device was made large enough to accommodate the Badgers and their USB leads. "Some of the challenges involved making the zoetrope light enough to be able to move," Brian says. "It also needed to be carried by one person, and robust enough to avoid being damaged by the public."

To control the entire device, Brian employed a Raspberry Pi 4 computer, using it to send screen updates over the USB connections. Another Raspberry Pi 4 is connected to a flatbed scanner and it allows animations created on a cell sheet to be scanned and uploaded to the zoetrope.

Warning!
Moving parts

Be careful with objects that spin at high speeds and do not touch them

A total of 15 Pimoroni Badger 2040s have been used. Each 2.9-inch e-ink display has a resolution of 296 × 128 pixels

Brian has designed the zoetrope to be as easy to operate as possible, and it includes a big red emergency stop button, controlled by Raspberry Pi Pico

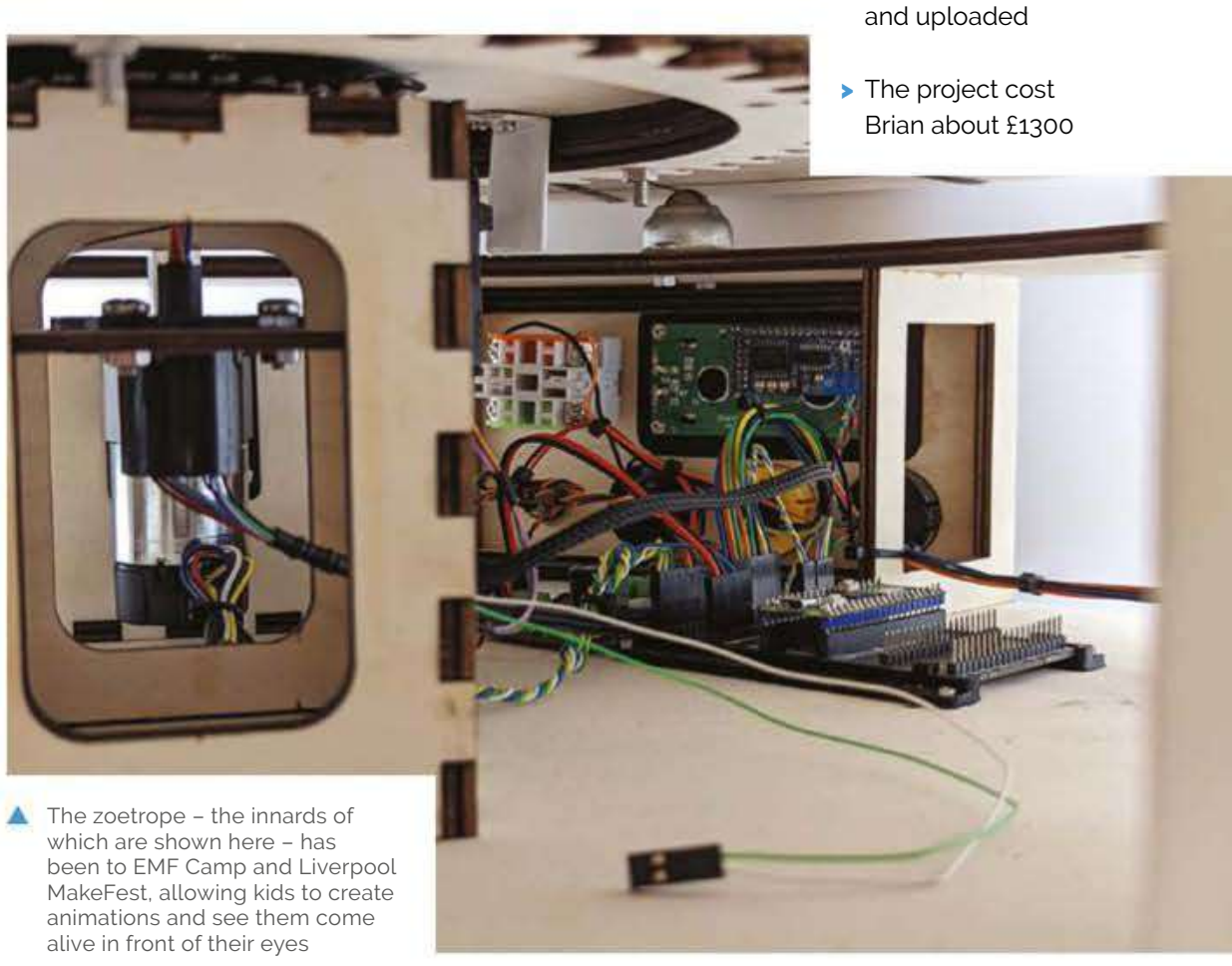
Quick **FACTS**

- This huge build uses e-ink displays
- It also involves laser-cutting plywood
- It requires Raspberry Pi Pico and Raspberry Pi 4 devices
- New images can be created and uploaded
- The project cost Brian about £1300

> Some of the challenges involved making the zoetrope light enough to be able to move

This means kids can get creative at events – or else press a touchscreen monitor and see past animations!

"The zoetrope has gone down a storm with children, and I'd like to thank Pimoroni for supplying me with 20 Badgers, Phil Howard, software developer at Pimoroni, for creating the custom firmware that enables images to be uploaded, and Brian Starkey for his help pasting together my code and knocking out a web user interface. This project would not have been possible without the support of the EMF art installation fund either."



▲ The zoetrope – the innards of which are shown here – has been to EMF Camp and Liverpool MakeFest, allowing kids to create animations and see them come alive in front of their eyes

Fireballs Aotearoa

A chance to expand meteor monitoring to New Zealand was made far simpler with the aid of Raspberry Pi. **Rosie Hattersley** reports

MAKER

Dr James Scott, Jeremy Taylor, Jim Rowe

Dr James Scott, a geologist at the University of Otago, and Jeremy Taylor set up the Fireballs Aotoreoa meteor tracking programme with input from Jim Rowe of the UK Fireball Alliance.

fireballs.nz

The excitingly named Fireballs Aotearoa (fireballs.nz) is an ambitious research project that aims to make good on the Global Meteor Network's aim of ensuring no meteor is undetected. The GMN's worldwide meteor and meteorite tracking endeavours already had sites spread across Europe and the US, but few in the Southern Hemisphere. With international news coverage of a fireball over southern England in March 2021 that was successfully tracked by citizen scientists, there has been a significant increase in the number of meteor cameras in the UK and beyond. A Fireballs Aotearoa outreach programme involving New Zealand-based astronomers and meteor researchers generated much excitement among would-be space scientists.

▲ The meteor camera network will monitor the skies over the whole of New Zealand

This online presentation given by Jim Rowe (**magpi.cc/findingmeteorites**) explains how a camera attached to a Raspberry Pi locks on to a fireball as it travels across the sky, focusing on the object itself and only processing data relating to its constantly changing location. Raspberry Pi has sufficient processing bandwidth to live-track and report the meteor's location. Since multiple cameras across a region track the same meteor's journey, it's possible to triangulate its final destination and work out with some accuracy where it must have landed, and potentially recover it, for further study, explains Jim.

> Raspberry Pi and Sony camera lenses have become the de facto hardware

New horizons

Raspberry Pi and Sony camera lenses have become the de facto hardware, lowering the cost of setting up a meteor camera to less than £200. Expanding coverage in Scotland and New Zealand is important for Jim Rowe of the UK Fireball Alliance. Jim was incredibly excited

Fireballs Aotearoa meteor cameras use Sony IMX291 image sensors to record 720p footage of meteor showers and other wonders in the night sky

Once the data is processed, Raspberry Pi uploads each set of data to a server at the University of Western Ontario

A Raspberry Pi 4 or 3B+ controls each setup, identifies the meteor within a frame, and saves only this part of each 25 fps image to its SD card

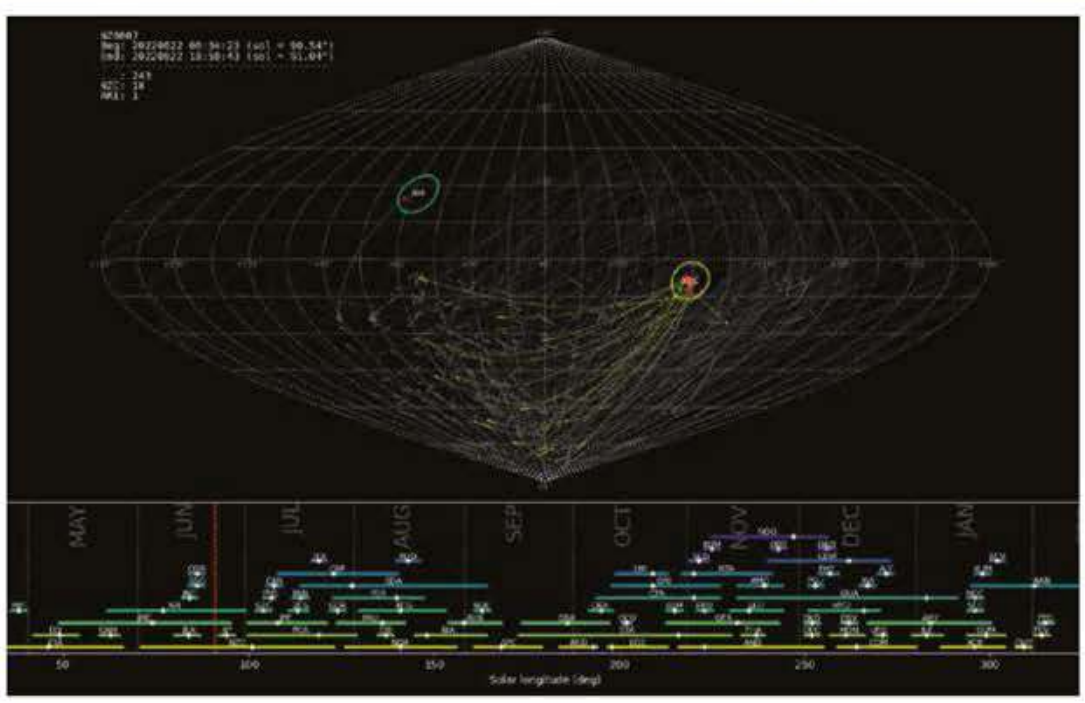


Illustration of the activity captured by the NZ0007 camera over the course of several months

Quick **FACTS**

- The New Zealand cameras capture meteors that would otherwise be missed...
- ...since most such cameras cover the northern hemisphere
- Access to the meteor camera data by schools is critical
- By involving them in planetary study, it's hoped...
- ...rural Otago students might be inspired to become astronomers

The project is already detecting plenty of meteor activity in the southern hemisphere, as this open-source meteor map reveals

when Dr James Scott of the University of Otago succeeded in getting funding for the first ten meteor camera kits for use in New Zealand, and helped acquire 20 more Raspberry Pi computers for meteor tracking and university use with a grant from the MBIE Curious Minds Participatory Science Platform. A key part of the funding pitch is the direct link with schools and their access to the project data. James says the goal is to record meteors on every clear night – a single camera in Invercargill picked up 114 meteorites crossing the sky one evening in early March – with students able to log in at any time to see what has crossed 'their' night sky and incorporate that in their schoolwork.

"Rather than simply recording pictures of meteors, the idea is to collect science-grade data that can inform researchers, capturing information about meteor orbits, frequency, flux, mass indices, source regions, and so on. This can be used to refine prediction models and help us learn more about parts of the solar system nearest to us. All the high-level data products from the GMN project are publicly released under CC BY 4.0 and updated every six hours so researchers can have access to near-real-time information," explains Jeremy Taylor (aka Tasmanskies) who has been "a driving force" behind the meteor camera builds.

UK Fireball Alliance's James Rowe's talk on meteor tracking showed how Raspberry Pi-powered cameras helped triangulate locations

Searching for a sky fall

Only nine meteorites have been discovered in New Zealand, and only the 1908 Mokoia meteorite in Tauranga was seen to fall. With the country having a land-mass larger than the UK, Dr Scott aims "to discover the next meteorite

▲ Multiple camera systems are cross-referenced to aid the recover of falling meteors

that comes into New Zealand through a citizen-led initiative." The meteor cameras are put together by students from the rocketry club at the University of Otago and installed at locations around New Zealand, creating the densest southern hemisphere meteor-tracking network. The network is already capturing plenty of activity. "Raspberry Pi is essential for calculating the meteor trajectory each camera picks up and determining the 'strewn' field where debris should have landed," says James.

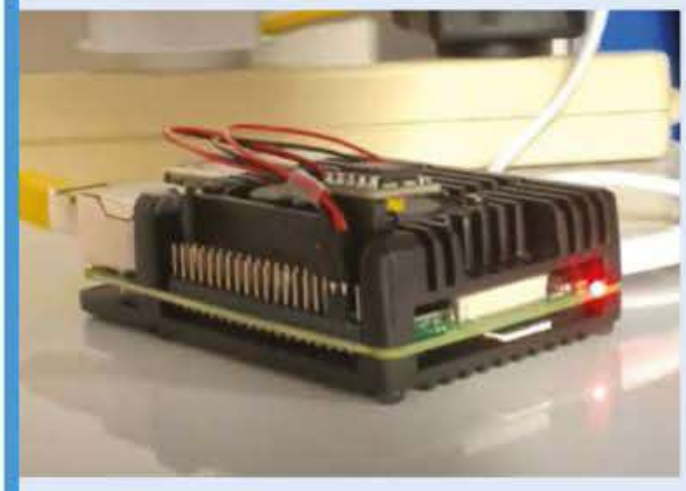
> Raspberry Pi is very easy to program and operate. We easily link to the boards at the schools via a remote connection

"Raspberry Pi is awesome; small enough to sit around unobtrusively, but powerful enough to control a night-sky camera and manipulate the data that it collects, such as generating stacked images of the duration of the night," enthuses James. "Raspberry Pi is very easy to program and operate. We easily link to the boards at the schools via a remote connection, which enables us to see the live stream, edit images, and access photos and videos. These are easily recompiled into time-lapses. It's just brilliant."

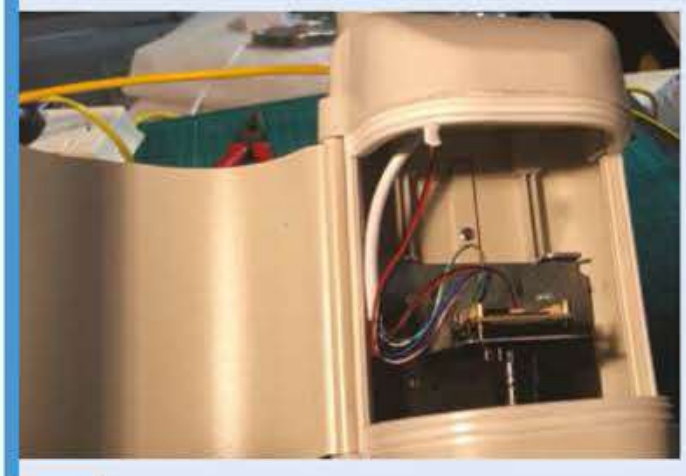
As a future development, Jeremy Taylor hopes it will be possible to install meteor cameras in Antarctica – with Raspberry Pi inside, of course!

Build a meteor camera

01 You will need a Raspberry Pi 3B+ or 4, HQ or equivalent camera, and a waterproof housing. Follow Jeremy's setup instructions provided at **magpi.cc/assemblingGMS**. Start by unscrewing the lens to remove the infrared filter, and cut off the camera mount nubbins so it sits flush.



02 Attach a heatsink and run a power supply to Raspberry Pi, since you will need it to be running constantly if you want the best chance of capturing meteor footage.



03 Connect everything together and place it inside a waterproof casing. Install somewhere there's a clear sky view, as with this Fireballs Aotearoa meteor camera at a school site in Otago, New Zealand.

LEGO Reaction Wheel Inverted Pendulum

Tackling a classic problem of control system dynamics with LEGO and Raspberry Pi. **Phil King** takes a balanced view

MAKER

Juha

"A Finnish middle-aged guy," Juha runs the Brick Experiment Channel on YouTube, dedicated to building and experimenting with LEGO Technic bricks. He has worked for over ten years as a software engineer.

magpi.cc/becyoutube

Keeping an inverted pendulum aloft is a popular challenge in control theory – which deals with the behaviour of dynamical systems – and requires precise motion control to prevent it from falling over. It's something maker and YouTuber Juha, of the Brick Experiment Channel, learned about in courses he did 20 years ago, which inspired him to try it with LEGO and Raspberry Pi.

Made of LEGO bricks, his inverted pendulum requires active control to stay upright. A 'reaction wheel', also made from LEGO, is mounted on the pendulum and attached to a motor controlled by a Raspberry Pi Zero 2 W. "Rotating the wheel will generate torque on the pendulum and thereby change the pendulum angle," he explains. "The rest is just measurement and computation."

Well, that's the theory, but in practice it's not quite so easy and, as shown in his YouTube video (**magpi.cc/legopendulum**), Juha had to alter his Python code repeatedly and make hardware adjustments to get the system to work to his satisfaction.

▲ A Raspberry Pi Zero 2 W runs the Python code that takes the sensor readings and controls the motor accordingly

Precision control

A gyroscope and accelerometer on a mini IMU (inertia measurement unit) board are used to measure the pendulum angle, while Raspberry Pi runs a control loop for filtering data and calculating PID (proportional–integral–derivative) controller outputs for adjusting the motor's speed and direction. Automatically calculating corrections based on feedback, PID is one of the most common control methods used in industrial and mechanical applications, such as in a car's cruise control system. It's also fairly easy to implement.

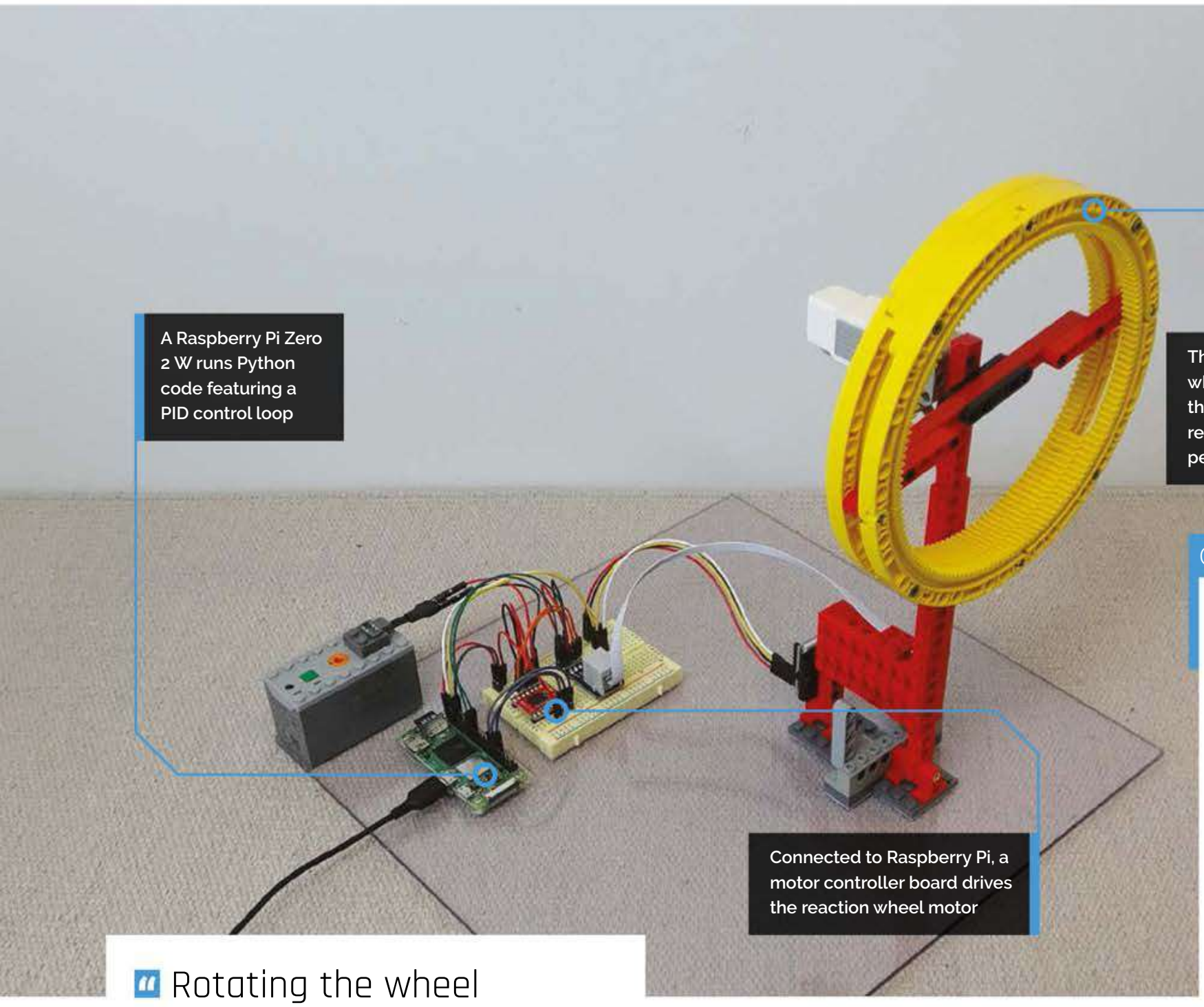
"Others have used LQR [linear–quadratic regulator] control for inverted pendulums," notes Juha, "but it looks too mathematical and difficult for me. As for tuning the PID parameters, I didn't have any approach other than 'try and see.' It got good laughs in the YouTube comment section as it looks so unprofessional."

While Juha opted to use a Raspberry Pi Zero 2 W for its fast bootup speed, "CPU load was only 5% running the control loop with a 1 ms interval, so it would work with a much less capable board." He even tried using a Pico, which "worked OK in terms of processing power, but then I realised I need to store tens of megabytes of log data for drawing nice graphs for the video."

Amazingly, before this project, Juha had never used a Raspberry Pi before and had minimal experience with electronics. "I had to figure out how GPIO works, what are pull-down and pull-up resistors, how I2C works, etc."

Keeping it up

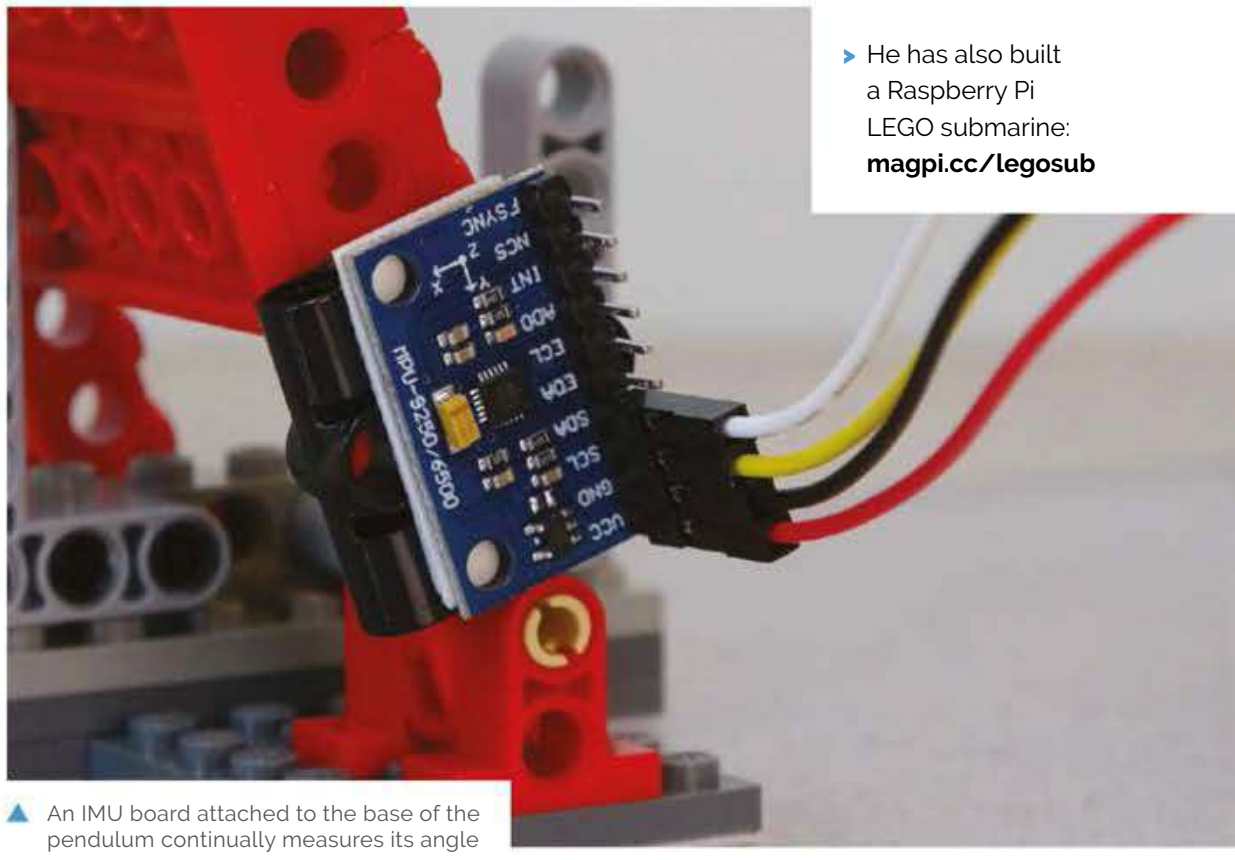
The next major challenge was getting the pendulum to stay upright for more than two seconds. "The problem was with the reaction wheel top speed limitation," says Juha. "There is a short time window for acceleration before the limit is reached, so you need to get past the top equilibrium point before that. A plain PID controller would just



> Rotating the wheel will generate torque on the pendulum and thereby change the pendulum angle

minimise angle error and keep the wheel rotating too fast. I read many studies of different inverted pendulums, until I found one paper that mentioned continuously changing the target angle for the PID controller. That solved it finally."

So, in the finished system, is it impossible to push the pendulum off balance so that it falls over? "No, not at all," replies Juha. "It will easily fall over if you push it. The controller will immediately try to compensate for the push by accelerating the wheel, but it can correct only for small errors. With a more powerful motor, higher top speed, and higher rotational inertia for the wheel, it would resist stronger pushes."



▲ An IMU board attached to the base of the pendulum continually measures its angle

Quick **FACTS**

- The project took Juha two months to build and perfect
- He says "it was painful to get it working"
- His Python controller code was rewritten several times
- The final program can be found at **magpi.cc/pendulumcode**
- He has also built a Raspberry Pi LEGO submarine: **magpi.cc/legosub**

Boost-Box 0.1

The antithesis of doom-scrolling, the Boost-Box could be just the thing to cheer you up. **Nicola King** discovers the joy of just watching happy things!

MAKER

Martin Spendiff & Vanessa Bradley

Martin is a mathematical modeller who left the UK for Switzerland, and a fan of FOSS and tech that serves users, rather than the people who made it. Vanessa is new to coding and a constant source of weird and good ideas.

veeb.ch

We just can't get enough of Martin Spendiff and Vanessa Bradley's makes – last month the Pico Railway Clock and, this time, we present the Boost-Box. The productive pair have upcycled an old 1970s Hanimex analogue film viewer and turned it into a YouTube viewing terminal – but they only use it to watch uplifting cheerfulness!

Martin came across the film viewer in a second hand store and began to feel nostalgic. "When I first went to uni a million years ago, we used terminals to do most of our work," he recalls. "That aesthetic is imprinted on my brain. I saw it and thought 'I need to do something with that'… but I wasn't really sure what I was going to do with it."

After applying some thought, he realised that it made sense to update it, but make its use related to its original purpose. "I liked the idea of it going from being able to show the contents of a single roll of film, to everything YouTube has on it. What is it, a billion videos or so?"

Ins and outs

So the duo set to work on what turned out to be a relatively straightforward build, and the first job was to substitute the original screen with an LCD replacement. "The main annoyance," says Martin, "was getting the screen I found with the right aspect ratio to fit into the frame. Luckily, I have watched plenty of videos of very patient people using metal files to shape things, so I settled down for an hour and made the aperture the right shape."

Next, they added a Raspberry Pi 4 and an ortholinear keyboard (with non-staggered keys). There are very few internal components in the original body of the viewer, as it's mainly empty space. The few parts that were moved to make space for a small speaker have been placed into a small bag inside the machine; this means the terminal could, should they so choose, be quickly turned back into a Super 8 viewer.

The project's Raspberry Pi 4 runs the Manjaro Linux operating system, while ytfzf (**magpi.cc/ytfzf**) is used as the command-line viewer tool. "The great thing about open source is that there is a huge array of tools that clever people have already built, so if you can 'glue' them together, you can build things quickly," enthuses Martin. "It's a pretty capable desktop machine. There are other command-line interface (CLI) tools that I've played with in the past (or use now) that would fit. NeoMutt (email) and gcalcli (Google Calendar) are two that spring to mind."

Practically perfect

The short YouTube video that Martin and Vanessa created to accompany this build is really worth a watch (**magpi.cc/boostbox**), as you'll see Boost-Box's spirit-lifting capabilities in action. Friends of the pair have given their feedback, and they seem to enjoy the retro look of the piece as much as anything, as Martin shares: " 'I love it, what is it?' is the common response, followed closely by, 'What is that keyboard?' "

The screen was replaced with a modern LCD to play videos

A 1970s Hanimex E300 Super 8 film viewer is used as the case for a vintage look

An ortholinear keyboard is connected to a Raspberry Pi 4 inside the case

Quick **FACTS**

- The Hanimex case cost around £25…
- …while the screen cost about £30
- You could connect it to bigger speakers using Bluetooth
- Martin has used an LED light power converter…
- …which powers Raspberry Pi and the screen

Martin describes the make as "not a hard project" and, as for future improvements to the Boost-Box… well, maybe it doesn't really need

> I'm avoiding doing more on it, as I like the idea of it just doing one thing

any. "For now, I'm avoiding doing more on it, as I like the idea of it just doing one thing," says Martin. "There are a few minor tweaks that I might make, but for now it's just sitting there playing 'Guinea Pig Olympics' every time the world gets a bit too much for me."

Frankly, who doesn't crave a shot of high-octane joyfulness on a regular basis – perhaps we all need to make our own Boost-Box?

These vintage film editors can be purchased relatively cheaply if you want to replicate the build

LED Sphere

Tom Verbeure has brightened up our day with his stunning sphere peppered with LEDs, as **David Crookes** discovers

MAKER

Tom Verbeure

Tom is a hardware engineer for Nvidia, who loves playing and occasionally blogs about all kinds of electronics projects.

magpi.cc/tomverbeure

There are many practical applications for a Raspberry Pi Pico microcontroller board, but it's important to have a heap of fun too! With his latest project, Tom Verbeure has literally had a ball, creating a NeoPixel-covered sphere that dazzles in more ways than one. After all, as Tom says: "LED cubes have been pretty popular, but spheres are a much harder problem, mechanically."

The idea for the project followed a friendly chat. "I was having a lunchtime discussion at work with my friend, Jens, a 3D printing wizard, about doing things with LEDs that hasn't really been done before," Tom recalls. After ordering a 3D printer – "and wanting to use it for something more than printing out an Iron Man helmet or a vase" – he put a plan into action. "One of my biggest requirements from the start was a lot of LEDs."

Bouncing ideas

By the time he got started, other makers were creating something similar. Jiří Praus unveiled his Freeform LED Sphere ("but it's constructed out of differently sized rings around a central axis, which I didn't want") and Whity created the Geodesic(k) RGB LED Spheres ("but 180 LEDs, and the LED density was too low because he used premade WS2812B PCBs").

A sphere element containing 21 LEDs. "You need to insert the LED through the sphere element, then try to fit each of its four metal leads through a very tiny 0.85 mm diameter hole," Tom explains

With research, he learned that it's mathematically impossible to distribute points uniformly across a sphere, but there were techniques which came close. "I loosened the requirement of uniform LED distribution somewhat and chose an icosahedron as the core internal structure," he says. "Its 20 axes of symmetry is sufficiently high.

"I also came to the conclusion that through-hole LEDs were the way to go if you want the surface of the sphere to be truly curved. You can adjust the length from the LED core to the PCB for each LED individually." He then spent more than a month using FreeCAD to come up with potential design ideas, starting with a central lattice made out of hollow triangles on which 20 PCBs and 20 triangular sphere elements were mounted separately.

On a roll

Printing and assembly issues caused a rethink. "The breakthrough came when I dropped the central frame, inserted LEDs from the outside into the shell, and mounted the PCBs firmly against the inside shell triangle," Tom explains. "I used magnets on each triangle side to form a self-supported structure. The more sphere elements snap together, the sturdier it becomes."

Eventually, he switched from FreeCAD's GUI to its embedded Python engine. He also settled on 21 LEDs for each of the 20 sphere elements, designing a custom controller PCB on which to mount them. A Raspberry Pi Pico formed the project's heart and Tom initially coded the project using MicroPython, later transitioning to regular C.

"I needed something that was small, didn't require a lot of power, had a lot of performance, was close to the metal, had the ability to drive

The controller board contains a dual 18650 battery holder, a power regulator module, Raspberry Pi Pico, ESP8266, and an MPU 6050 inertial/gyroscope sensor (for orientation aware light effects)

Although an ESP8266 wireless networking microchip was used, Raspberry Pi Pico W means this isn't necessary any more

Custom triangle PCBs are connected to each of the sphere sections. Each supports 21 LEDs and it contains 21 Surface Mounted Device (SMD) resistors and capacitors

Quick **FACTS**

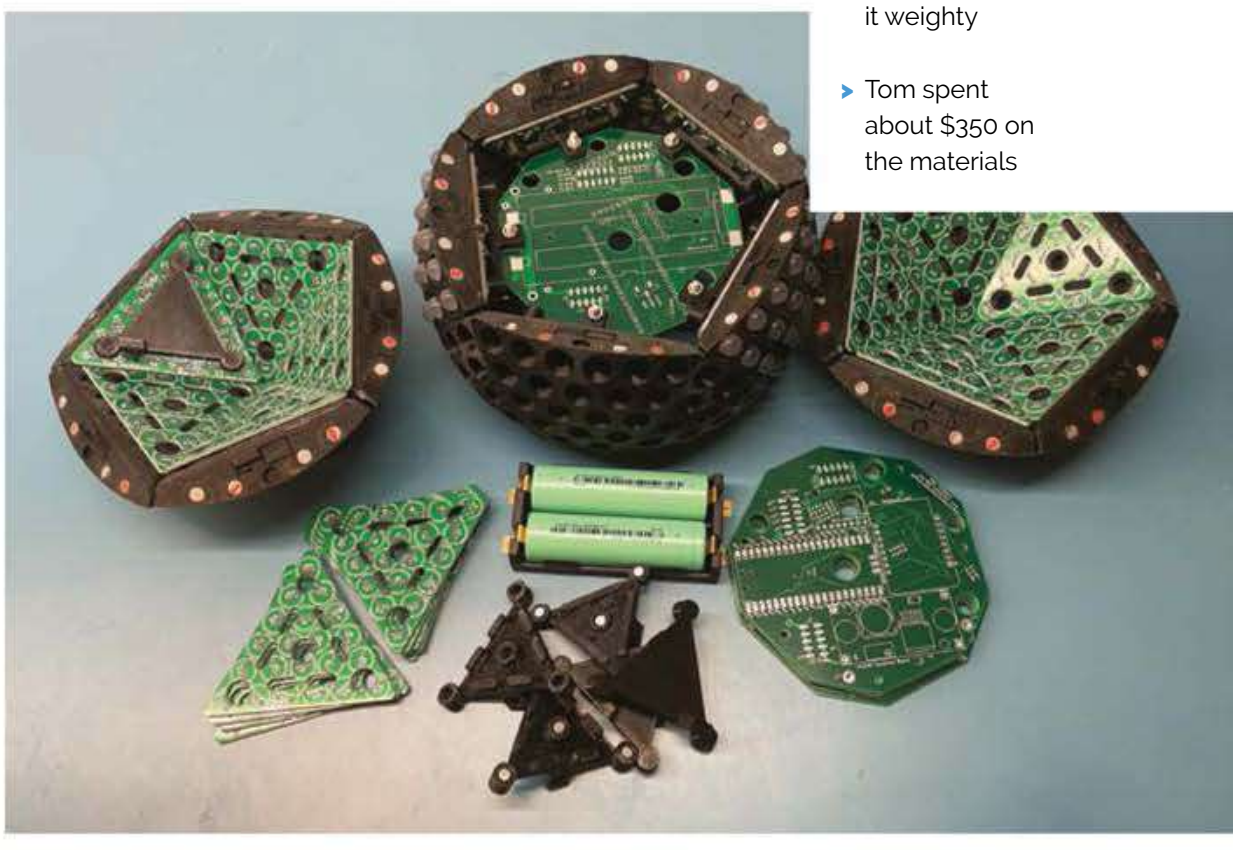
- The 20 sphere elements contain 420 LEDs overall
- Assembling the sphere elements took 20 hours
- The project uses 400 tiny neodymium magnets
- The magnets and Li-ion 18650 batteries make it weighty
- Tom spent about $350 on the materials

> I used magnets on each triangle side to form a self-supported structure

WS2812 LEDs, and had very good documentation and examples," Tom notes. "The Raspberry Pico scored high on all those points."

The result is impressive. "The LEDs are not just a part of a sphere element, but part of the whole sphere, with a unique (x,y,z) coordinate assigned to it," Tom says. It's also battery-operated, making it entirely mobile.

Tom has enjoyed the process so much, he's on a roll, continuing to make improvements. "Right now, I'm dealing with a fair bit of instability due to loose wires," he says. "If I were to do this again, I'd probably use JST connectors instead of plain vanilla pin headers and connectors, because they make a very reliable link."



Tom said working with magnets was a pain during construction. "If you make one polarity mistake while supergluing them, it can be impossible to remove them later"

SUCCESS STORY magpi.cc/success

Brompton

Demand for Brompton's famous folding bicycles has soared. Raspberry Pi has helped the company meet challenging production targets at its London manufacturing facility. By **Rosie Hattersley**

Brompton folding bicycles are a British urban icon. Invented, designed, and manufactured in London, their popularity has grown at pace, and in recent years annual sales have topped £100 million as Brompton has become the go-to solution for commuters seeking a smooth and swift end-to-end trip. "We've tried to make people aware that there's a product that might make their life a bit happier, give them a bit more freedom," says CEO Will Butler-Adams. Brompton's factory systems have evolved to support the demands of increasing volume and an expanded product range, and Butler-Adams describes their Raspberry Pi-powered factory setup as slicker than any other he has seen.

THE CHALLENGE

Brompton first introduced Raspberry Pi in 2013 on, what was then, a line of nine production stations. The computers – the original Raspberry Pi 1 Model B – were used to track which models were built at which station and when. The following year, Brompton expanded capacity to 16 stations in order to meet rising demand, but this still wasn't enough: they needed to up production further, all while continuing to track units at every station, and introducing further monitoring to help meet increasingly demanding targets. "The only way to do that," concluded senior software engineer Kane Tracey, "was Raspberry Pi."

THE SOLUTION

Tracey says that there are over 100 Raspberry Pi computers inside Brompton's factory, deployed in solutions ranging from pre-assembly-line and production-line monitoring to air quality control and more. A number of models are in use, including Raspberry Pi 4 and earlier boards.

Raspberry Pi boards on production stations scan the serial number of each bicycle, tracking it as it progresses through the assembly process. The computers use the scan data to monitor and support production in multiple ways: for example, staff are provided with unit-specific instructions via connected displays, a laser-etching machine automatically produces the correct plate design for the type of bike under assembly, and LED indicators

> Raspberry Pi plays a crucial role in Brompton's assembly processes

help staff to meet time targets. In addition to scan data, production-line Raspberry Pi boards capture data from torque controllers, from user input, and more. Key performance indicators are shown on display screens for ease of monitoring.

"We have this philosophy here at Brompton now that if we need to capture data anywhere on the factory floor, we throw a Raspberry Pi at it," says Kane Tracey.

WHY RASPBERRY PI?

The Brompton team have investigated other single-board computers, aiming to identify the most cost-effective platform that could offer the power and flexibility needed for low-friction deployment into a broad range of applications. They concluded that Raspberry Pi offered both superior performance and better value than other boards on the market. "We've done a lot of research," reflects Tracey. "There are plenty of other small credit-card-sized PCs that come out, and they don't match Raspberry Pi. We've tested them and there's nothing on the market that compares. Not just for performance, but also for value."

As production volume has increased at Brompton, strict time targets have become ever more important to their operations, Tracey emphasises. "The Raspberry Pi now is critical in our manufacturing. We have to stick to the take time, we have to stick to how many bikes we get out. And this is what the Raspberry Pi does for us: it helps us reach those targets."

THE RESULTS

Each Brompton bicycle is a masterpiece of perfection in design and technology, reflecting the pride Brompton assembly operators take in ensuring that every machine is constructed to the very highest possible standards of quality.

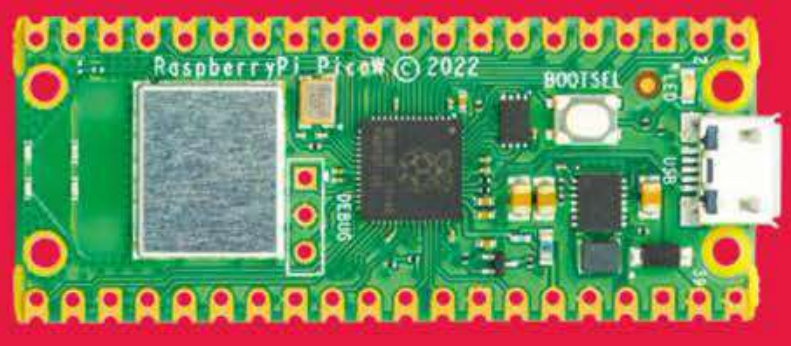
Raspberry Pi plays a crucial role in Brompton's assembly processes, ensuring critical elements such as the application and capture of the correct torque – essential for the safety of the rider – as well as capturing every other aspect of the build process through each station of the assembly line.

Butler-Adams is clear about the impact of Raspberry Pi at Brompton. "It gives us insight, allows us to adjust, it gives us traceability, it helps training. And I have visited a lot of factories: I've never seen a system as slick as the system we've created ourselves with the humble Raspberry Pi."

SUBSCRIBE TODAY FOR JUST £10

Get 3 issues + FREE Pico W

+



Subscriber Benefits

- **FREE Delivery**
  Get it fast and for FREE
- **Exclusive Offers**
  Great gifts, offers, and discounts
- **Great Savings**
  Save up to 35% compared to stores

Subscribe for £10

- Free Pico W
- 3 issues of The MagPi
- Free delivery to your door
- £10 (UK only)

Subscribe for 12 Months

- Free Pico W
- 12 issues of The MagPi
- Free delivery to your door

£55 (UK) £90 (USA)
£80 (EU) £90 (Rest of World)

Subscribe by phone: **01293 312193**

Subscribe online: **magpi.cc/subscribe**

Email: **magpi@subscriptionhelpline.co.uk**

Subscribe for £10 is a UK-only offer. The subscription will renew at £30 every six months unless cancelled. A free Pico W is included with a 12-month subscription in USA, Europe and Rest of World. (No Raspberry Pi Pico W with £5 Rolling Monthly Subscription).

LEARN ELECTRONICS WITH PICO W

EXPLORE ELECTRONIC COMPONENTS AND START BUILDING YOUR OWN PROJECTS WITH THE NEW WIRELESS RP2040-BASED MICROCONTROLLER.

BY LUCY HATTERSLEY

Raspberry Pi Pico W is an incredible development board, built around the powerful RP2040 microcontroller.

On both sides of Pico W sit GPIO (general-purpose input/output) pins. There are 40 of them in total. These pins are used to interface Pico W with the wider world. You use them to wire up electronics components, attach Pico W to computer equipment, and build personal projects.

This is the real joy of Pico W (and Raspberry Pi computers). These pins are where code interfaces with the real world. You can use Pico to automate your home, control robots, or manage household appliances. Or, you can simply mess around with buttons, buzzers, sensors, and build fun widgets and gizmos.

Also, thanks to Pico W's wireless capabilities, you can now interact with online services, pulling in data from the internet via APIs (application programming interfaces), and storing information wirelessly.

PICO W WITH HEADERS

This Pico W has been populated with two headers (small pins soldered to the GPIO holes). This enables it to be attached to the breadboard for prototyping. You can also solder wires and components directly to the GPIO holes, but most newcomers start with headers. Pico WH, sold with a header attached, will be available to buy soon.

JUMPER LEADS

Jumper leads are used to connect the various components to one another and to the GPIO pins on Pico W, forming a circuit.

COMPONENTS

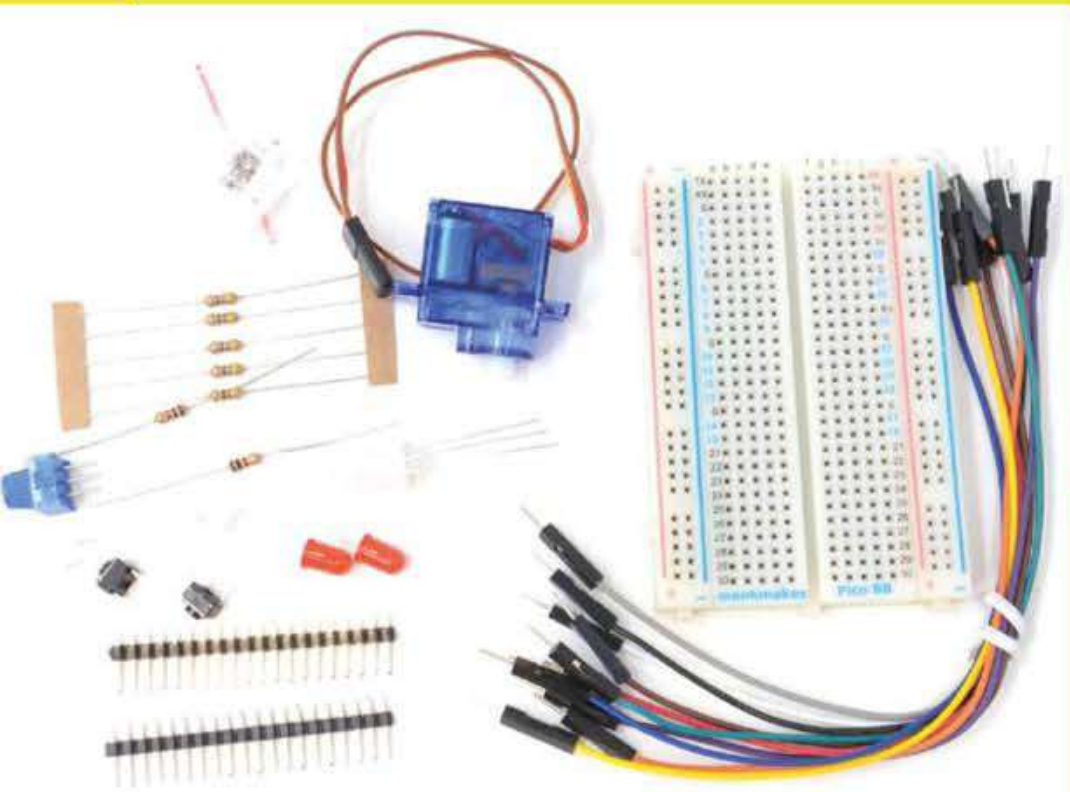
Our board here has two components: a small LED (light-emitting diode) and a resistor (which is used to reduce the voltage and protect the LED). Around the board are some other components you will frequently encounter, such as buttons, light and distance sensors, and a small display.

BREADBOARD

A breadboard is a white slab of plastic populated with holes. Each hole in a row is connected, enabling you to connect wires and components by putting them in a hole in the same row.

ELECTRONIC KITS & PARTS

DISCOVER PICO W ELECTRONICS EASILY WITH THESE STARTER KITS



Electronics Kit for Pico (Lite)

MonkMakes • £15/$20

Coming from *The MagPi* regular Simon Monk, this kit features ten well-documented projects in a downloadable 68-page book. It also contains a breadboard, LEDs, buttons, a phototransistor, and a piezo buzzer, plus jumper leads and resistors. Projects include a light meter, thermometer, and fader. A great starter option.

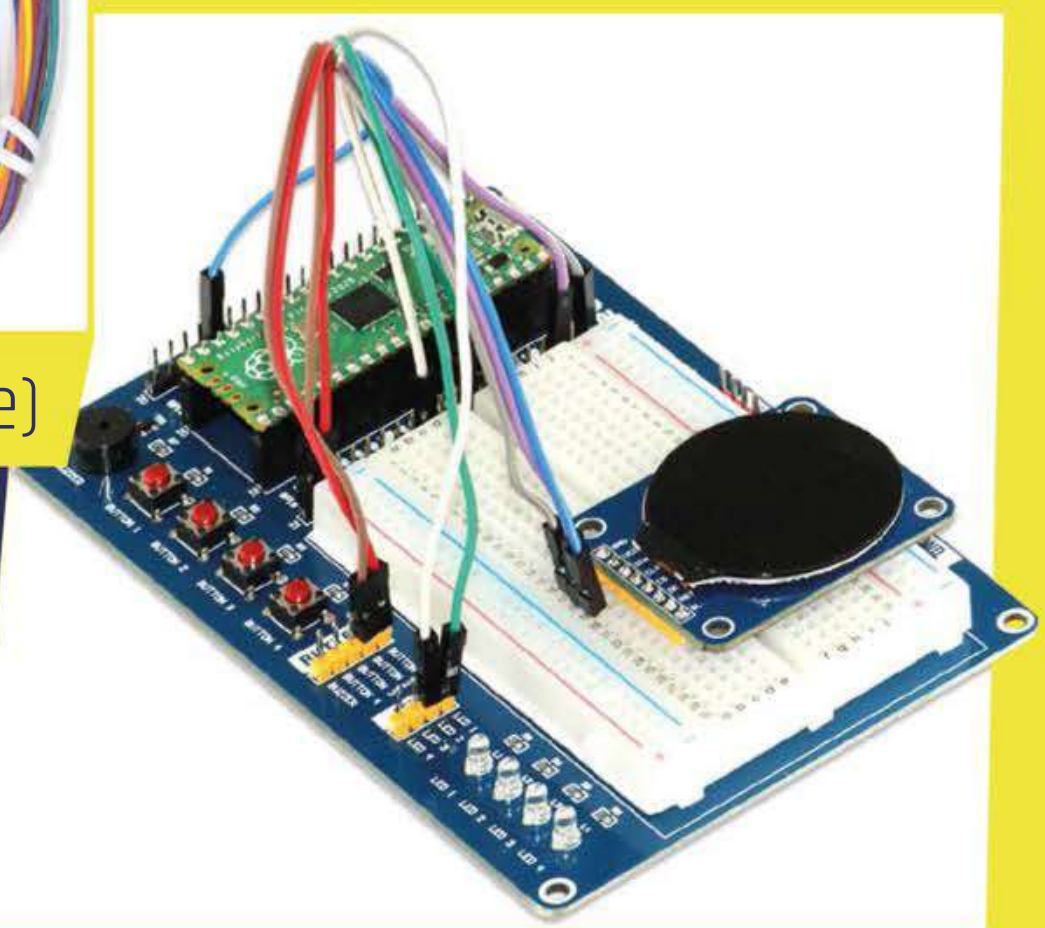
> magpi.cc/eleckitpicolite

Pico Breadboard Kit

SB Components • £14/$17

This breadboard kit contains a half-sized breadboard mounted onto a board with four LEDs, four push-buttons, and a buzzer. The rest of the GPIO pins are broken out with a detailed pinout guide. One of the most useful prototyping kits around.

> magpi.cc/picobreadboardkit



Kitronik Inventor's Kit for Raspberry Pi Pico

Kitronik • £30/$36

A physical booklet accompanies this prototyping plate with breadboard. It also has a more comprehensive list of components including a seven-segment display, LED Zip stick, SG90 servo, motor and fan blade, buzzer, and a wide range of LEDs and leads.

> magpi.cc/inventorskit

Inventor 2040 W

Pimoroni • £35/$43

This all-in-one board comes with a Pico W already pre-soldered on to it. It's a versatile board that is packed with connections for attaching components. You can add two motors to the JST-SH connectors, hook up a speaker, attach servos, and a battery, and it comes with twelve RGB LEDs to play with. Plus, two Qw/ST connectors that enable you to attach just about anything. It's a really interesting piece of equipment for all manner of creations.

> magpi.cc/inventor2040w

PICO W RESOURCES

Get Started with MicroPython on Raspberry Pi Pico

MicroPython and Pico

Created by the team at Raspberry Pi Press, this book is packed with official information on how to program Pico with the MicroPython language.

magpi.cc/picobook

Pico Documentation

Raspberry Pi's website has a stack of documentation designed to introduce you to all the details that make up Pico.

magpi.cc/picodocs

Pico Forum

If you are stuck with a problem, or want to see what folks are saying about Pico W, then bookmark Raspberry Pi's dedicated Pico forum page.

magpi.cc/picoforum

RUN A WEB SERVER ON RASPBERRY PI PICO W

CONNECT PICO W TO A BREADBOARD AND USE A WEB CONNECTION TO TURN A LIGHT ON AND OFF

MAKER

Alasdair Allan

Alasdair Allan is Head of Documentation at Raspberry Pi.

raspberrypi.com

In *The MagPi* issue #119, we introduced Raspberry Pi Pico W. Based around Raspberry Pi's own RP2040 microcontroller, Pico W brings 802.11n wireless networking to the family of Pico boards.

This means that your Raspberry Pi Pico W can now talk to the network, but also that the network can talk back to it; and you can run a web server on your Pico W to allow you to control things remotely.

You'll need to attach GPIO headers to Raspberry Pi Pico W and use a breadboard with an LED light for this tutorial. Set everything up as shown in **Figure 1**.

You'll Need

- Raspberry Pi Pico **magpi.cc/pico**
- Raspberry Pi (or laptop) **magpi.cc/products**
- Breadboard, LED light, resistor, jumper leads **magpi.cc/electronicskit**

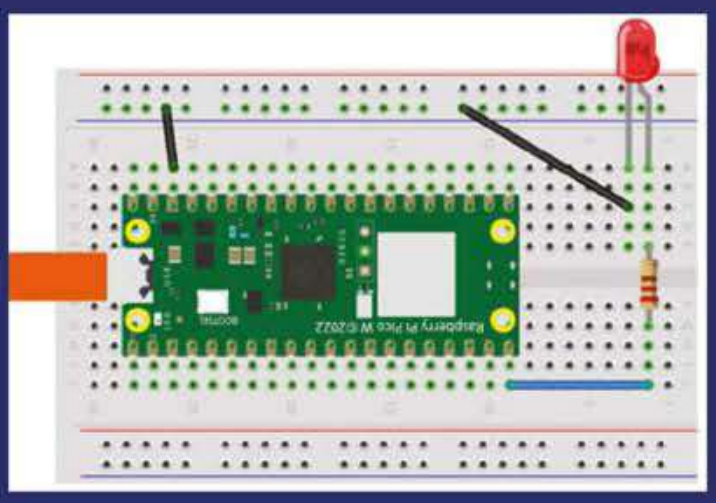


Figure 1 Connecting your Raspberry Pi Pico W to a LED

01 Installing MicroPython

The fastest way to get MicroPython is to download the pre-built release binary from the Documentation pages (**magpi.cc/micropython**).

Then go ahead and push and hold the BOOTSEL button, and plug your Pico W into the USB port of your computer. Release the BOOTSEL button after your Pico W is firmly connected, and it will mount as a mass storage device called RPI-RP2. Drag and drop the MicroPython UF2 file onto the RPI-RP2 volume. Your Pico W will now reboot. You are now running MicroPython, and you can now access the REPL via USB Serial.

02 Controlling a LED via the web

When you're writing software for hardware, turning an LED on, off, and then on again is typically the first program that gets run in a new programming environment. Learning how to blink an LED gets you half way to anywhere. We're going to do exactly that, via a web browser.

We are in fact going to build a RESTful(ish) web server to control our LED.

We've chosen to attach an external LED to GP15 of our Raspberry Pi Pico W, but you could just as easily use the on-board LED for testing things out.

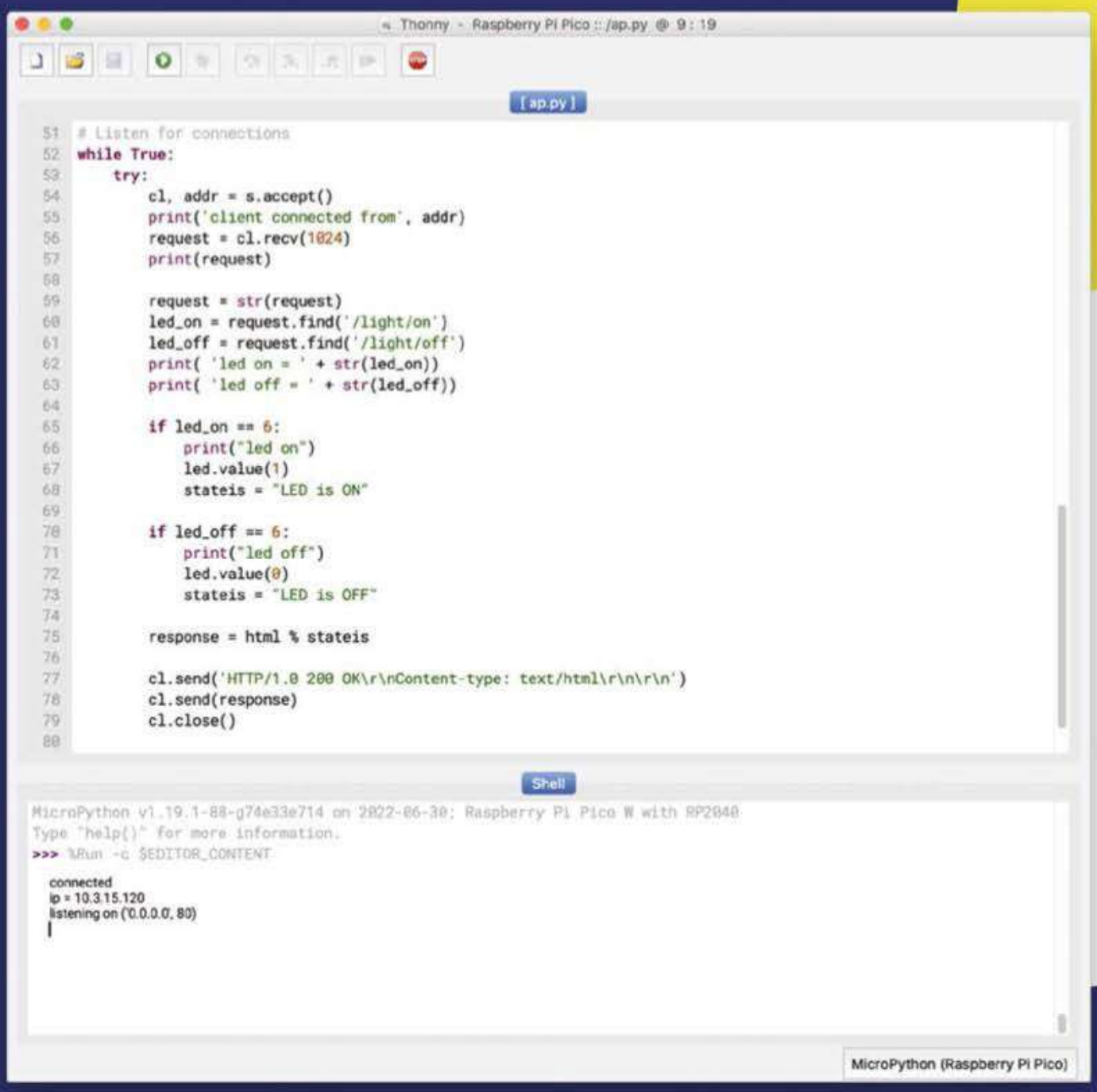
The LED light is connected to Pico W's GPIO pins via jumper leads and a resistor

GPIO header pins are soldered to Pico W and the board is attached to an electronics breadboard for circuit prototyping

Top Tip

Getting started

More information on setting up Pico W can be found in this online tutorial: **magpi.cc/setuppicow**.



Running our Python script on Pico W from the Thonny editor

03 Start Thonny

Open Thonny, and upload the **webserver.py** Python script to your Pico W. If you haven't used MicroPython and Thonny before, full instructions on how to do that can be found in the Raspberry Pi Pico Python SDK book (**magpi.cc/picopythonsdk**).

Make sure to replace the `ssid` and `password` with the name and password for your own wireless network at home, and then hit the button to run the script on your Pico W.

04 Light on

After your Pico W connects to your wireless network, you should see the IP address for your board appear on the REPL inside the Thonny shell window.

To control the LED, you can open up a web browser and go to **http://10.3.15.120/light/on** to turn the LED on, and **http://10.3.15.120/light/off** to turn the LED off again.

You should substitute your IP address, which for most home networks will probably be in the 192.168.1.X range, for the one shown here.

05 Going further

This example lets you remotely turn an LED on, and then off again. However, we can also extend this example to add buttons to the web page we're serving to allow you to control the LED directly from a web interface rather than by using a RESTful server – which, after all, is more suited to programmatic use rather than working from a web browser. Alternatively we can go further and reimplement our server, so that rather than blocking, it will operate asynchronously.

More information about connecting your Pico W to the web can be found in the online documentation and the *Connecting to the internet with Raspberry Pi Pico W* book: **magpi.cc/internetpicow**.

Top Tip

Soldering

You'll need to solder GPIO pins to Pico W. See Raspberry Pi's guide to soldering: **magpi.cc/soldering**.

webserver.py

DOWNLOAD THE FULL CODE:
magpi.cc/github

Language: **Python**

```
import network
import socket
import time

from machine import Pin

led = Pin(15, Pin.OUT)

ssid = 'YOUR NETWORK NAME'
password = 'YOUR NETWORK PASSWORD'

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)

html = """<!DOCTYPE html>
<html>
    <head> <title>Pico W</title> </head>
    <body> <h1>Pico W</h1>
        <p>%s</p>
    </body>
</html>
"""

max_wait = 10
while max_wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    max_wait -= 1
    print('waiting for connection...')
    time.sleep(1)

if wlan.status() != 3:
    raise RuntimeError(
'network connection failed')
else:
    print('connected')
    status = wlan.ifconfig()
    print( 'ip = ' + status[0] )

addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]

s = socket.socket()
s.bind(addr)
s.listen(1)

print('listening on', addr)

# Listen for connections
while True:
    try:
        cl, addr = s.accept()
        print('client connected from', addr)
        request = cl.recv(1024)
        print(request)

        request = str(request)
        led_on = request.find('/light/on')
        led_off = request.find('/light/off')
        print( 'led on = ' + str(led_on))
        print( 'led off = ' + str(led_off))

        if led_on == 6:
            print("led on")
            led.value(1)
            stateis = "LED is ON"

        if led_off == 6:
            print("led off")
            led.value(0)
            stateis = "LED is OFF"

        response = html % stateis

        cl.send('HTTP/1.0 200 OK\
r\nContent-type: text/html\r\n\r\n')
        cl.send(response)
        cl.close()

    except OSError as e:
        cl.close()
        print('connection closed')
```

Exploring electronics with a Pico Breadboard Kit

Raspberry Pi Pico W brings physical computing and the internet together and it's never been easier. Let's learn the basics by making a weather indicator

MAKER

PJ Evans

PJ is a writer, software engineer and general tinkerer. He can currently be found replacing all his old microcontroller projects with Raspberry Pi Pico Ws.

@MrPJEvans

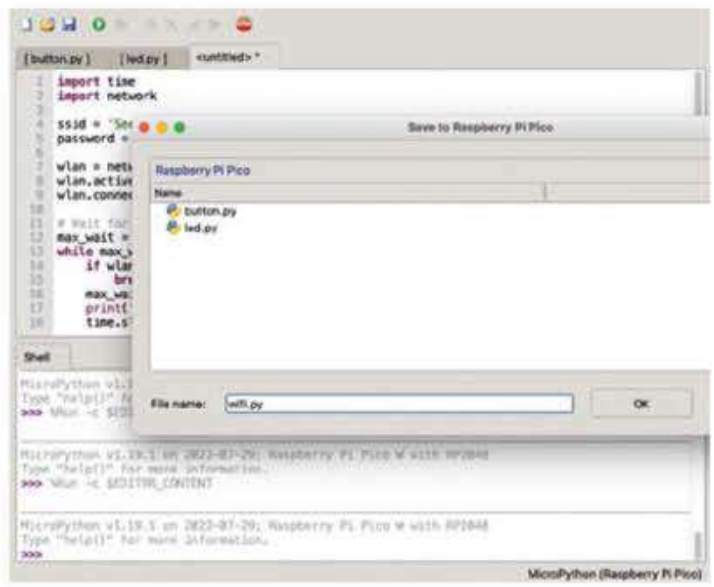
If the new Raspberry Pi Pico W's wireless LAN capability has got you eager to start making but you're not sure where to start, you're in the right place. We're going to take a step-by-step look at simple components for inputs and outputs, connect them to Pico W, and then fetch data from the internet and display them. We're going to do this without any of the difficulty of soldering, or even handling components, by using the SB Components Pico Breadboard kit. This PCB (printed circuit board) comes pre-populated with buttons and LEDs to make your introduction to electronics as simple as possible.

You'll Need

- SB Components Pico Breadboard Kit **magpi.cc/picobreadboardkit**
- 2 × 20-pin headers (if not already there) **magpi.cc/picoheaders**
- 10 × Female-female jumper cables **magpi.cc/jerkyjunior**
- Micro-USB to USB cable suitable for your computer

01 Pico preparation

A great facet of Raspberry Pi Pico W is its support for different languages. By uploading different 'firmwares' (low-level code that translates for the RP2040 CPU), we can write programs in different ways. We're using MicroPython, a microcontroller flavour of Python, which greatly simplifies writing code for Pico W. It also includes everything we need to connect to the internet. To download and install the latest version of the MicroPython firmware, follow the instructions here: **magpi.cc/micropython**.



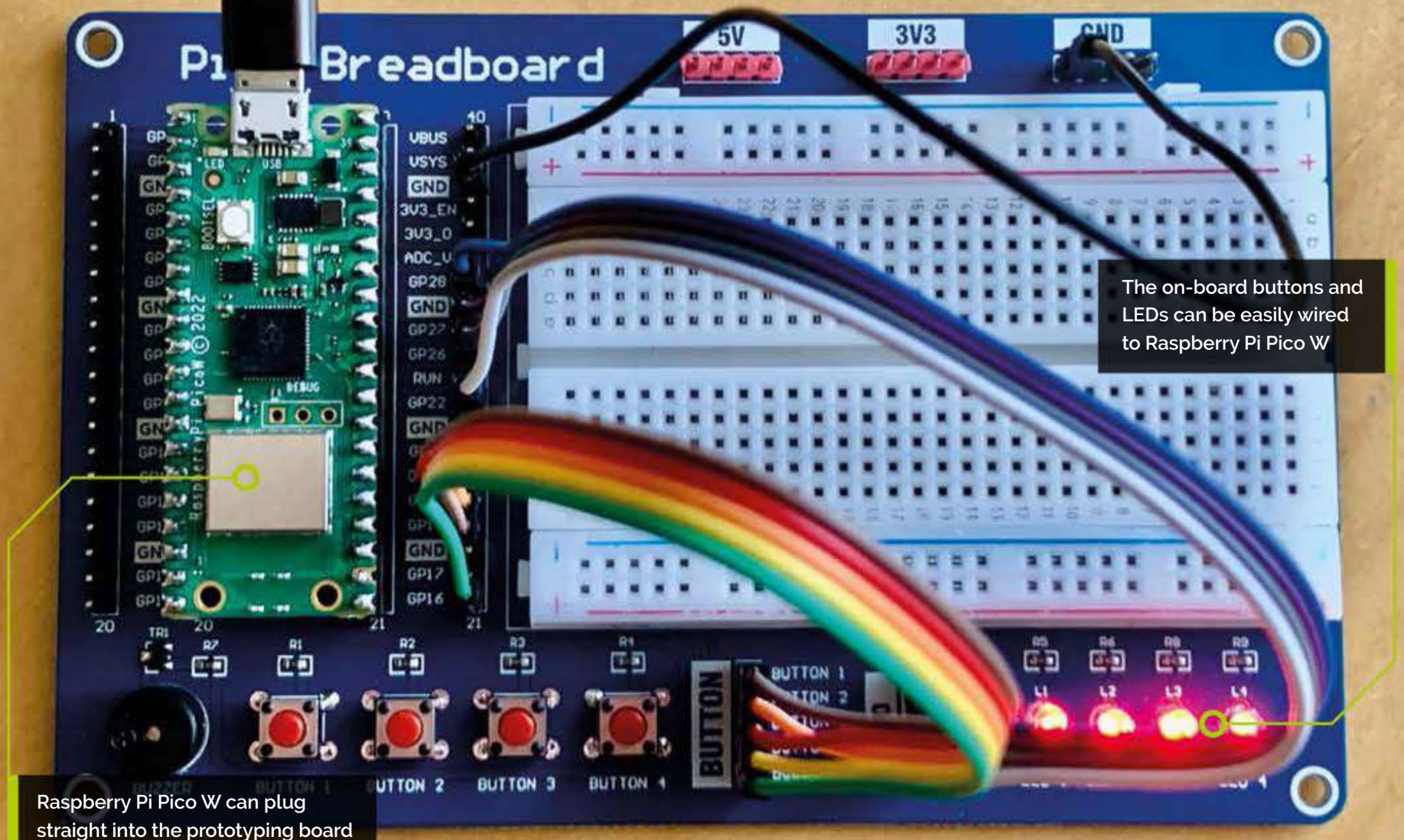
You can write code as files and upload them directly to Pico W. To have code run instantly when Pico W is plugged in, just name it **main.py**

02 Get ahead

To connect to the Pico Breadboard Kit, you will need to have headers soldered onto your Pico. If you do not have a Raspberry Pi Pico H (H for headers), then you need to buy a header kit and solder them on yourself. These need to be facing downward so the smaller part of the header is poking through the top of Pico's PCB on the RP2040 chip side. If soldering yourself, remember to be careful and start by soldering each end pin of the header block, then check everything is level. If not, you can melt the solder to move them into place, then solder all the ones in between.

03 Choosing a development environment

To write code for Raspberry Pi Pico W, you need to use a computer. Nearly any modern operating system will do, including Raspberry Pi OS. It's possible to write code using a simple text editor, but it's a lot easier (and faster) to use an IDE (integrated development environment). Don't be put off by the fancy name – this is a text editor that understands what a Pico W is and can help transfer programs. On Raspberry Pi OS we recommend Thonny, but you can also use Visual Studio Code with the Pico-Go extension. More info: **magpi.cc/gettingstartedpico**.



The on-board buttons and LEDs can be easily wired to Raspberry Pi Pico W

Raspberry Pi Pico W can plug straight into the prototyping board with all GPIO pins accessible

04 Testing time

It's time to check everything is working. Connect your Raspberry Pi Pico W to your computer using a USB cable. Open up Thonny and look in the bottom right-hand corner. It should say something like 'Python 3.7.9'. Click on this and, if Pico W has been recognised, you'll see 'MicroPython (Raspberry Pi Pico)' as an option. Select this and you'll see a welcome message on the bottom half of the screen. Click to the right of the '>>>' prompt and type:

```
print('Hello')
```

...followed by pressing **RETURN**. If you see 'Hello' displayed in response, you've just run your first program on your Pico W!

05 Understanding LEDs

Have a look at the SB Components prototype board. On the bottom-right are four LEDs (light-emitting diodes). These are one of the most common components used when beginning electronics, as we can make them light up and that's cool! LEDs can be a little naughty and draw too much current if left unchecked, causing permanent damage. To stop this happening, we need an in-line resistor to limit the flow of current. See the little black and silver squares above the LEDs? They are 330 Ω resistors already in place that are perfect for the job, so we can wire everything up without worrying about damaging our LEDs.

> We are using the RP2040's built-in 'pull-down' resistor circuit which solves this problem

06 Understanding buttons

On the left-hand side of the prototyping board are four buttons (and a buzzer, but we'll get to that). Each button creates a circuit when the button is pressed down. By wiring these to your Raspberry Pi Pico W, you can detect when the button is pressed. Buttons can be tricky for a microcontroller to handle, as the input is so sensitive it can give inaccurate readings; you can even trigger it by putting your finger next to it. To prevent this, we are using the RP2040's built-in 'pull-down' resistor circuit which solves this problem. When it comes to coding, you'll see how we make use of it.

07 Get wired

It's time to assemble our circuit. Carefully insert the disconnected Raspberry Pi Pico W into the socket on the prototype board, with the USB end at the top. The LEDs and buttons connect to Pico W using jumper cables. Between the buttons and the LEDs you'll see two yellow sets of headers, clearly labelled. The jumper cables need to run from these to the GPIO pins on Pico W. If you're wondering where the ground connection is, look at the top-right of the board. The GND header must have one wire connected to any of the GND pins on Pico W. Follow the wiring table (overleaf) carefully.

Top Tip

Keep it in order

LEDs or buttons not working in the right order? Check the wiring, it's really easy to get things the wrong way around.

08 Light up the LEDs

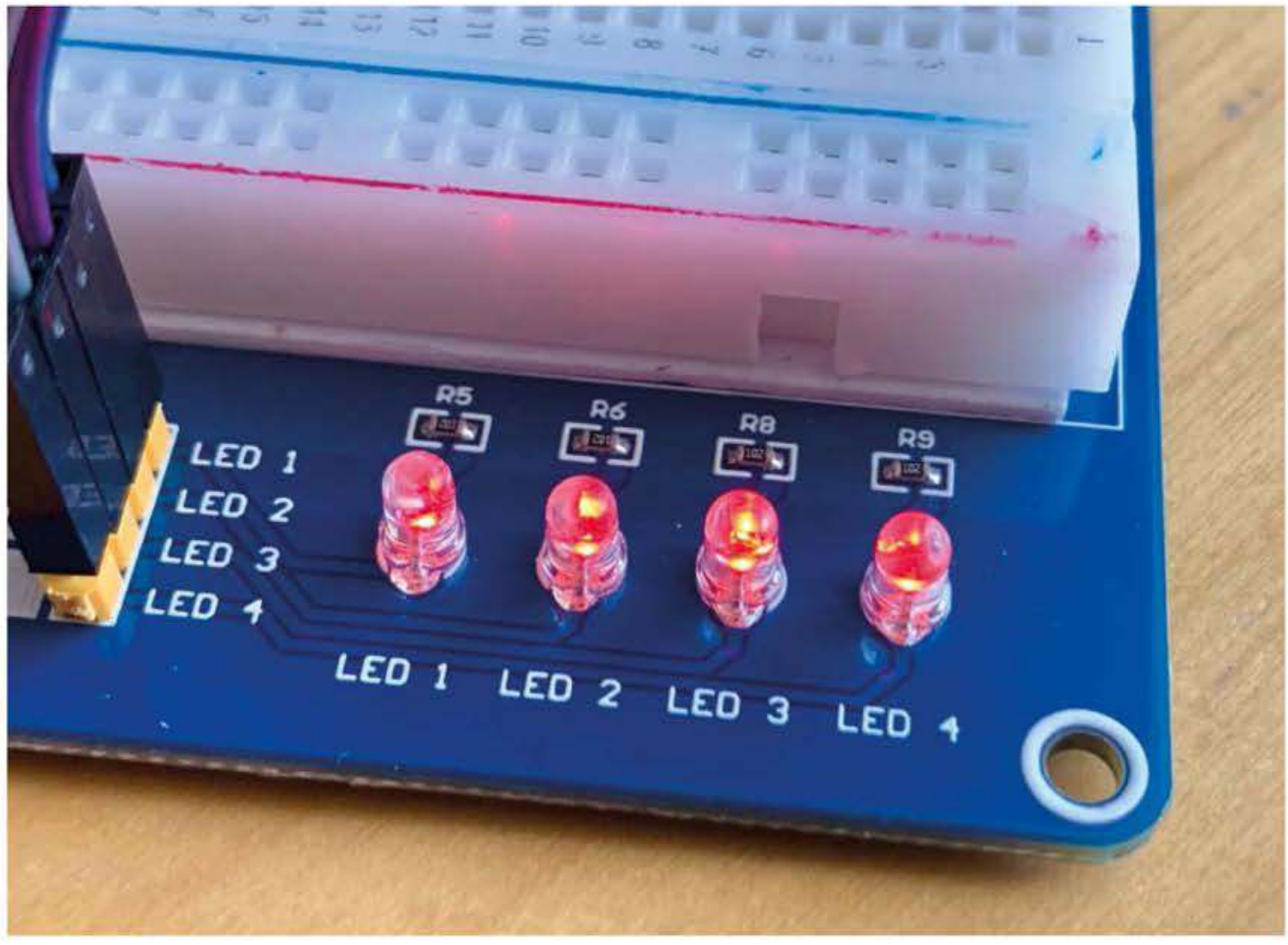
Having checked all your wiring carefully, connect Raspberry Pi Pico W to your computer. In Thonny, type the **leds.py** code listing into the upper window, and then click the 'Run' icon. When prompted, ask to save it on the Pico W and name it **leds.py**. The code will now be uploaded to Pico W. Do you see the LEDs coming on one-by-one? The code starts by telling Pico W which GPIO pins are connected and what they are for (in this case - output). Then we go into a loop: toggling the state of each pin, then waiting a second. If you want, have a play with the sequence or see if you can change timings.

09 Push the button

To check the buttons, create a copy of the file you created in Step 8, and call it **buttons.py**. Remove the block starting `while True:`, then add the contents of the **buttons.py** code listing. Save and run the code. Try pressing the buttons one by one. Each one should now toggle its equivalent LED. This code uses 'event' or 'interrupt' handlers, blocks of code that run when a GPIO pin changes state. When a button is pressed, the code runs and changes the state of the LED. This is a fundamental part of physical computing. You are taking external input (the button) and creating output (the LED).

10 Simon says

You've now seen how Raspberry Pi Pico W can use code to respond to inputs by



LEDs require in-line resistors to prevent them drawing too much current. Here, those resistors are already provided

With a few lines of code, Pico W can connect to your wireless network and then to the internet

creating outputs. We could have wired the buttons directly to the LEDs to create a similar effect (and using the breadboard, you can try that!), but Pico W adds logic that would be hard to implement in raw circuitry alone. To demonstrate this, download **memory.py** from **magpi.cc/memorypy**. This is an extension of the code that turns our button script into a memory game. Run the code on your Pico W, and see if you can remember the sequence of LEDs by playing them back on the buttons. Don't forget to review the code and see how it works!

> If you're wondering where the ground is, look at the top-right of the board

11 Get online

Now we have built our circuit, tested it, and played a game, let's look at what makes Raspberry Pi Pico W so special. For our weather project, we need to connect to the internet, so let's start with that. Create a new file called **wifi.py** and add the contents of the **wifi.py** listing (overleaf). Replace the `ssid` and `password` values with those for your own network. Now run the code on your Pico W using Thonny. Watch the console output and within a few seconds you should get an IP address announcement, meaning you're on the internet!

12 A key step

We're going to get some weather info to display on our prototype board. We'll use **openweathermap.org** to supply information using an API call. This is just like getting a web page, except the information is returned in a way computers can easily understand (in this case JavaScript Object Notation, or JSON). Sign up for a free account on the site and, once logged in, go to

leds.py

DOWNLOAD THE FULL CODE: magpi.cc/ledspy

Language: **Python 3**

```
001. from machine import Pin
002. import utime
003.
004. # Make sure these are the pins connected to your LEDs!
005. leds = {
006.     1: Pin(28, Pin.OUT),
007.     2: Pin(27, Pin.OUT),
008.     3: Pin(26, Pin.OUT),
009.     4: Pin(22, Pin.OUT),
010. }
011.
012. # Loop through the LEDs toggling each one then sleeping a
     second
013. while True:
014.     for i, (k, led) in enumerate(leds.items()):
015.         led.toggle()
016.         utime.sleep_ms(1000)
```

buttons.py

DOWNLOAD THE FULL CODE: magpi.cc/buttonspy

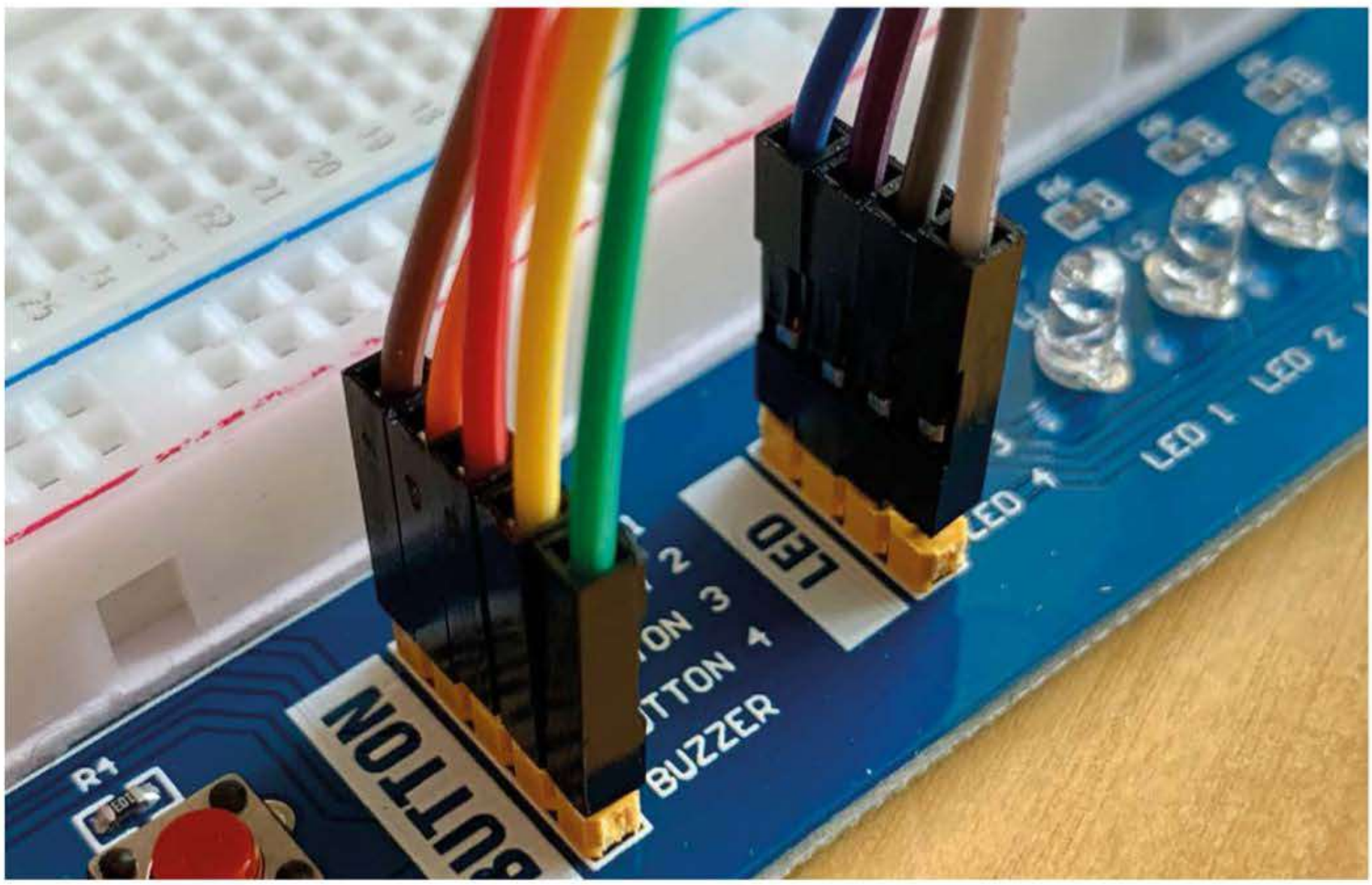
Language: **Python 3**

```
001. # Remove the while True: block and replace with this
002. def button_handler(pin):
003.     button_pressed = int(str(pin)[4:6]) - 17
004.     print(str(button_pressed))
005.     leds[button_pressed].toggle()
006.
007. for gpio_number in range(18, 22):
008.     button = Pin(gpio_number, Pin.IN, Pin.PULL_DOWN)
009.     button.irq(trigger=Pin.IRQ_RISING, handler=
     button_handler)
```

'API Keys'. One will have already been created for you (although it can take an hour or two to start working). Think of this as a password allowing you to access the service. Take a copy – you're going to need it soon.

13 Talk about the weather

We're going to make a request for the current weather. Download **weather_1.py** from



These handy jumper cables make connecting the LEDs and buttons to Pico W easy and safe

Here are the connections you need to make between Pico W and the button and LED connectors. Be careful and don't forget the GND!

magpi.cc/weather1py, then replace the three new variable values at the top with the API from the previous step and your desired latitude and longitude. Don't know these? Just enter 'lat and long for town' in a search engine, and you'll get the answer. The ones in the code are for the Raspberry Pi Foundation in Cambridge. Run the code on your Raspberry Pi Pico W as before, and watch the console output. Here, we use the urequest library to request information from the API server.

	Button	LED	GND 1-4
GP28		1	
GP27		2	
GP26		3	
GP22		4	
GP21	1		
GP20	2		
GP19	3		
GP18	4		
GND			Any

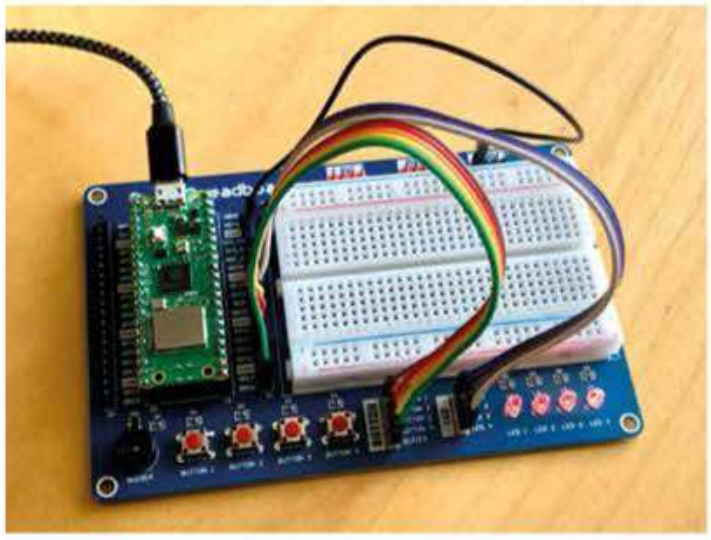
14 Hooking up the buttons

We've got four buttons, so let's pick out four key pieces of information. Each time we press a button, a request will be made to the API and the code will extract a useful piece of data from the request. We're going to ask for temperature, windspeed, rain, and air quality index. This code is a little longer, so download **weather_2.py** from **magpi.cc/weather2py** and transfer it to your Raspberry Pi Pico W as before. Run the code and press each button. Watch the output in the console as you press the buttons.

> Enter 'lat and long for town' in a search engine, and you'll get the answer

15 Lightening

We don't have a screen (although you can add one if you want!), just four lights to show the data. What we'll do is divide the results into ranges and light the appropriate amount of lights. For instance, If it's really hot, all four LEDs will illuminate. Download **weather_3.py** from **magpi.cc/weather3py** and run it. Raspberry Pi



You could use a standard breadboard and components, but the Pico Breadboard Kit makes it all a lot easier

Pico W will download the JSON data and extract our four data points and display them using the LEDs. Feel free to change the ranges if you wish.

16 Buzzin'

Calls to the internet can never be fully relied on to succeed. There are many things that can go wrong, from your internet connection being down to the API server having problems. We can catch these errors and signal to the user that there's a problem. The next version of our code (**weather_4.py** from **magpi.cc/weather4py**) creates a short buzz on successful calls and a longer buzz if something went wrong. Keep getting errors? More details will be logged to the console.

17 Put it all together

Let's bring the buttons and the LEDs together. Download our final code version, **weather_5.py** from **magpi.cc/weather5py**, and run it up as before. Now when you press each button, you can get an idea of whether it's raining, sunny, hot, or windy! Take some time to walk through the code to see how we hunt through the data, and see what changes you can make! If you would like to run this independently without a computer attached, just rename this file to **main.py**. Any file of that name will run automatically when power has been applied to Raspberry Pi Pico W.

To get the weather data you need an API key. You'll find it on the OpenWeatherMap user page like this

18 Make it your own

In this tutorial we've learnt how to control LEDs, listen for button presses, and combine those with internet data. Feel free to alter the code to show different things. Maybe you could periodically check the API and sound the buzzer when it's raining? This is just the beginning! For such a low-cost device, the capabilities of Raspberry Pi Pico W go much further than switches and lights. You can add all kinds of sensors, screens, and even motors with the right kit. Get an electronics kit and use the breadboard to add more features. There are endless tutorials out there to help you along. Be curious and have fun!

Top Tip

More data

OpenWeatherMap offers different APIs, many for free, so it's worth exploring what other data you could get, such as UV warnings.

wifi.py

DOWNLOAD THE FULL CODE: magpi.cc/wifipy

Language: **Python 3**

```
# Based on code by Pete Gallagher
# https://www.petecodes.co.uk/
import time
import network

ssid = "<Your Wifi Network Name>"
password = "<Your Wifi Password>"

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)

# Wait for connect or fail
max_wait = 10
while max_wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    max_wait -= 1
    print('Waiting for connection...')
    time.sleep(1)

# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('Network connection failed')
else:
    print('Connected')
    status = wlan.ifconfig()
    print( 'IP Address = ' + status[0] )

# Important to tidy up the connection
wlan.disconnect()
```

MagPet – code a Python virtual pet

Virtual pets are back! At least here in the magazine, as **Rob Zwetsloot** revives a nineties phenomenon on a Raspberry Pi

MAKER

Rob Zwetsloot

Rob is *The MagPi* Features Editor, and he sometimes fancies himself a game developer when he's not playing games.

magpi.cc

Didya see, didya hear, didya know that virtual pets are still going strong today? Your Tamagotchis, Digimons, and weird alien-themed knock-offs are still around with improved functionality. They're still stuck in their little plastic bodies though, so we thought it was high time we made one from scratch that you can play on your Raspberry Pi desktop. No magical crest required.

Make sure you're on the latest version of Raspberry Pi OS on your Raspberry Pi, and update all the software. Alternatively, you can make this on another computer, as long as you install the Pygame library.

You'll Need

- Latest Raspberry Pi OS
- Game sprites
- Pygame library (on other systems)

01 Get your art

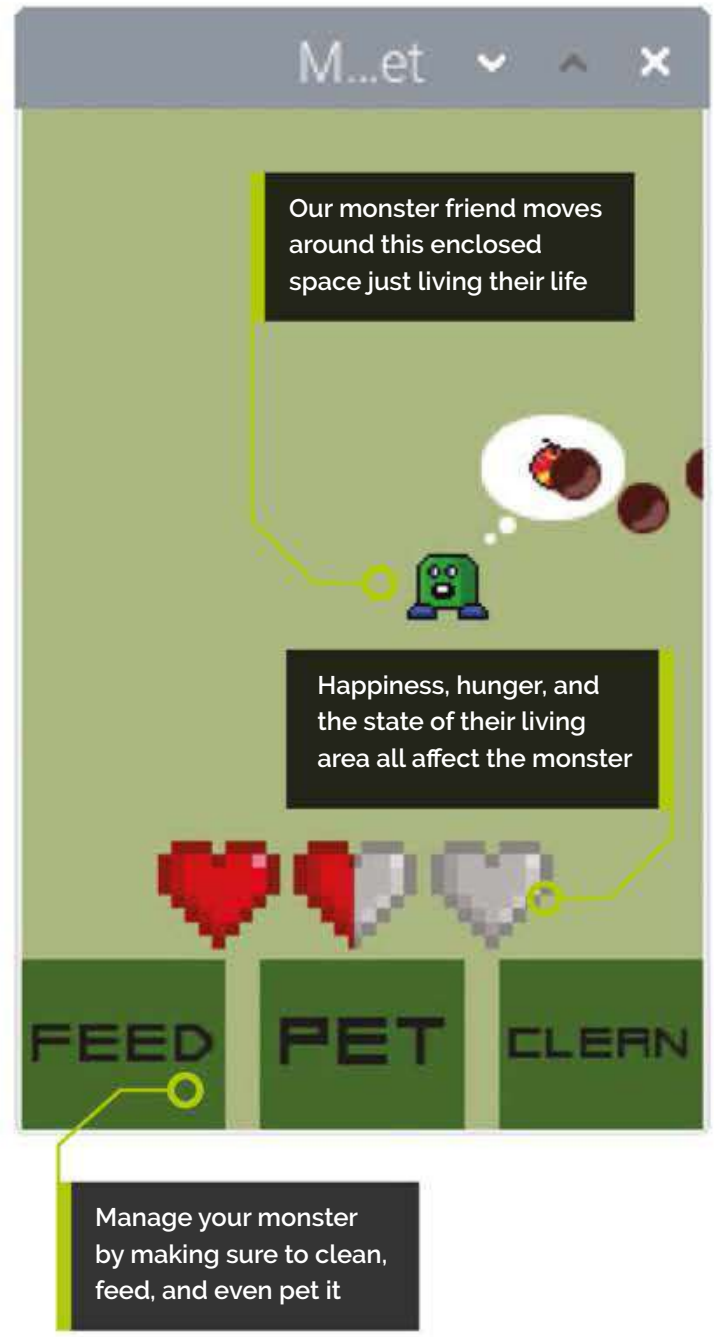
We've provided code and art on our GitHub at **magpi.cc/magpet**. However, our images are really just placeholders to make sure it all worked.

You'll need at least one image for your pet (ours has a basic two-frame animation cycle), different heart images for happiness, a graphic for when your pet is hungry, a picture of poop, and we also used graphic buttons we made ourselves. Save them in a directory where your Python code is.

Using `pygame.image.load`,you can set the image file to be a variable name, making calling upon it easier in the code.

02 Basic parameters

For our version of a virtual pet, we've only imported Pygame (which does include a lot of functions of its own) and the random module for part of the movement cycle.



The game screen will be 200 pixels by 300 pixels wide in our code. However, if you have large sprites, you may want to increase this.

We've made the background colour the same as a classic Tamagotchi with RGB, and decided to put the pet in the centre of the screen to start. We've also created some global parameters for health and happiness, walk cycle stage, etc., so that the screen updates properly along the way.

03 Move your pet

Our pet is going to move on its own. We'll handle the direction later on but, for now, we have two functions: `pet` and `movement`.

Movement uses cardinal directions translated as numbers (1 = North, 2 = East, etc.) which is selected in the main game code. We move the creature ten pixels in those directions, but also check to see if we're at a boundary so that the pet does not move any further.

Using these co-ordinates, we set the location of the pet in the next frame of the game – we also cycle between the different images that make up the walk cycle.

04 Handle health

What is the goal of this game? To keep our pet happy and healthy. On screen, three hearts are used to show how happy the pet is, with each heart being made up of 20 happiness points behind the scenes. We've created a series of `if` statements to build up the heart graphics, for full, half, and empty hearts.

As the pet relieves itself, it will deposit waste around the screen. The function `poopxy` keeps track of where it's gone, and we're very careful to make sure only one appears at a time. Each time a new poop appears, its co-ordinates get appended to a tuple.

```
import pygame
import random

# Game info

display_width = 200
display_height = 300

bg_colour = (160, 178, 129)

pet_x = 100
pet_y = 100

# Variables for the pet

hunger = 2
happiness = 20
waste = 0
wastexy = ()
button_press = 0
pet_counter = 0
walk_cycle = 0

# Images, including two images for pet idle animation

pet_1 = pygame.image.load("sprites/pet1.png")
pet_2 = pygame.image.load("sprites/pet2.png")
full_heart = pygame.image.load("sprites/fullheart.png")
half_heart = pygame.image.load("sprites/halfheart.png")
empty_heart = pygame.image.load("sprites/emptyheart.png")
hungry = pygame.image.load("sprites/hungry.png")
poop = pygame.image.load("sprites/poop.png")
clean_button = pygame.image.load("sprites/clean.png")
feed_button = pygame.image.load("sprites/feed.png")
pet_button = pygame.image.load("sprites/pet.png")

# Game functions
# Pet location

def pet(pet_x,pet_y,game_display):
    global walk_cycle
    if walk_cycle == 0:
        game_display.blit(pet_1, (pet_x,pet_y))
    else:
        game_display.blit(pet_2, (pet_x,pet_y))
```

Figure 1 Lines 1 to 35 cover the first two steps. Here, we get everything set up for the rest of the code

For protoyping software that needs graphics, you'll find just about anything you'll need on **OpenGameArt.org** for free

05 Game loop

Each loop of the game, our pet goes through a cycle. It will get hungrier, need to relieve itself more, and also will lose happiness if you're not paying it attention. We've also set it up so happiness is affected by hunger and cleanliness, so make sure to keep your pet fed and clean. When our pet is at a certain level of hunger, a little thought bubble appears to prompt you to feed it.

The `button_pressed` function checks to see where your mouse was when it clicked. This method of tracking pixels is a very basic way to press buttons in Pygame.

06 Main loop

We start the main loop by getting the rest of Pygame set up. This includes a clock to make sure each frame passes at a certain point, creating

Top Tip

What's in a name

Tamagotchi is a portmanteau of tamago/egg and tomodachi/friend in Japanese. They're shaped like eggs after all.

```
# Pet movement.
# There are checks so it won't move beyond the boundaries

def movement(move, game_display):
    global pet_x, pet_y
    if move == 1:
        pet_y -= 10
    if move == 2:
        pet_x += 10
    if move == 3:
        pet_y += 10
    if move == 4:
        pet_x -= 10
    if pet_x < 10:
        pet_x = 10
    if pet_x > 190:
        pet_x = 190
    if pet_y < 10:
        pet_y = 10
    if pet_y > 190:
        pet_y = 190
    pet(pet_x, pet_y, game_display)

# Happiness as displayed by hearts
# Each heart is worth 20 points

def hearts(game_display):
    global happiness
    if happiness < 1:
        game_display.blit(empty_heart, (60,215))
    if happiness > 0 and happiness < 15:
        game_display.blit(half_heart, (60,215))
    if happiness > 14:
        game_display.blit(full_heart, (60,215))
    if happiness < 35:
        game_display.blit(empty_heart, (80,215))
        game_display.blit(empty_heart, (100,215))
    if happiness > 34 and happiness < 55 :
        game_display.blit(half_heart, (80,215))
    if happiness > 34:
        game_display.blit(full_heart, (80,215))
    if happiness < 65:
        game_display.blit(empty_heart, (120,215))
    if happiness > 64 and happiness < 85:
        game_display.blit(half_heart, (120,215))
    if happiness > 84:
        game_display.blit(full_heart, (120,215))
```

Figure 2 Step 03 covers the last few lines of **Figure 1**, and also the movement part until line 68. Happiness is shown using the hearts function, explained in Step 04

the game display, and importing all the variables we need.

> We decided that it wouldn't just move randomly each frame

Now the full `while` loop starts. The screen is filled with colour, the buttons are placed on the screen, and the mouse co-ordinates are set to 0.

As this part is being called for each frame, it's best to keep as little out of it as you can to make sure your game runs as fast as possible. For this script, the global parameters and unchanging display aspects were placed before the `while` loop for this reason.

Top Tip

Size can matter

The size of your screen and sprites need to match in some way – it's easier to resize images out of Python after all.

07 Handling inputs

In this project, we have two kinds of inputs: a mouse click, which we're using to press the buttons on screen, and also the exit button on the Pygame window.

The `pygame.QUIT` event is fairly simple. If the X on the window is pressed, Pygame will stop. `QUIT` is the 'event' Pygame receives when you click on the X.

As for the mouse, we wait until the button goes up, hence `MOUSEBUTTONUP` is the event. This means it won't count you having the mouse button depressed as multiple events over several frames if you don't click fast enough. You may have seen variations on this with how buttons work in GPIO Zero.

08 Movement philosophy

Here's where we set the cardinal directions of the pet. We decided that it wouldn't just move randomly each frame, and it would in fact have a 60% chance to carry on in any direction it started. This is where the random module comes in. In action, this means our pet will move more naturally.

The cardinal direction and the `game_display` variable are sent to the `movement` function from

```
# Poop location

def poopxy(waste,pet_x,pet_y,game_display):
    global wastexy
    if int(waste) > len(wastexy):
        wastexy[((int(waste))-1)] = ((pet_x + 5), pet_y)
        prev_waste = int(waste)

    for i in wastexy:
        game_display.blit(poop, wastexy[i])

# Hunger and happiness cycle

def pet_cycle(pet_x, pet_y, game_display):
    global hunger, happiness, waste
    if hunger < 10:
        hunger += 0.2
    if hunger > 7:
        game_display.blit(hungry, ((pet_x + 25),(pet_y - 30)))
    if happiness > 0:
        happiness -= 0.05
        if waste > 5:
            happiness -= 0.4
        if hunger > 5:
            happiness -= 0.2
    hearts(game_display)
    if waste < 5:
        waste += 0.1
    if int(waste) > 0:
        poopxy(waste,pet_x,pet_y,game_display)

# Button locations for press

def button_pressed(mousex, mousey):
    global button_press
    if mousey < 190:
        return 0
    else:
        if mousex >= 0 and mousex <= 60:
            return 1
        elif mousex >= 70 and mousex <= 130:
            return 2
        elif mousex >= 140 and mousex <= 200:
            return 3
        else:
            return 0
```

Figure 3 Waste is managed as shown in Step 04, and from 106 to the end are the functions that handle your interactions and how your pet lives as shown in Step 05

Step 3. We pass on `game_display` so that the function can update it properly.

09 Button juggling

There are three buttons on this pet, which means there are four states to keep track of. We've tried to keep it simple and understandable by labelling the buttons 1–3, and no button being 0. One thing we did was to make sure we couldn't just hammer the Pet button to increase happiness. A timer is set so that you have to wait five frames before using it again.

As each frame is rendered separately, this means if you reset hunger and waste, the images for them will not be rendered in the following frame.

> We're planning to connect ours to web APIs

10 Game Over...?

How do you lose the game? Well, you need to take care of your pet, so if your pet gets as hungry as it can be (10), fills up its space with waste (5), and loses all happiness (0), it will be game over. On original Tamagotchi devices, they would run away. In our case, the game will just stop and you'll have to start again.

11 Actual play

If your code has made it this far, it's time to update the frame. All the important movement and location functions are called, the cycle is updated, and everything is rendered. As each frame goes by, this gives the illusion of life, and limited time to keep at it.

One important part of the end of the loop is `clock.tick()`. It's used to set a frame rate, with a lower number meaning fewer updates. As virtual pets of old would have very limited movement, we found that 2 simulated a good, slow-moving pet. You can always change it depending on the speed of your hardware.

Finally, the main function is run.

DOWNLOAD THE FULL CODE:
magpi.cc/magpet

Figure 4 This block of code handles Step 06 (144 to 158), Step 07 (160 to 166), Step 08 (168 to 177) and finally Step 09 for the rest

Figure 5 The game is played out through this last bit of code, described in Steps 10 and 11

12 Improvements

There's plenty you can do to upgrade this. Better graphics, better buttons, and a tighter adherence to the boundaries as well for starters. You could also add other features like your pet going to sleep for a while, or have it properly run away when it's game over. You could have a status screen with age, and even name it.

We're planning to connect ours to web APIs in the future to see if the internet can properly look after a virtual pet. We don't have high hopes though.

Part 01

Build a Raspberry Radio

Create your own virtual radio station, with a DJ that reads out the news and weather and announces your songs before they play

MAKER

Sean McManus

Author of *Mission Python*, *Scratch Programming in Easy Steps*, and *Raspberry Pi For Dummies* (with Mike Cook). Get free chapters at Sean's website.

sean.co.uk

We love radio, but don't you ever wish you had more control over the playlist? With Raspberry Radio, every song is your request. You give it a collection MP3 files, and the virtual DJ plays them at random, telling you something about each track. After a few songs, it's time for the news and weather. In this tutorial, you'll see how to download the news headlines using an RSS feed and access your local weather through an API. Next issue, we'll get the DJ talking and cue the music. You can download the code at **magpi.cc/newsreader**.

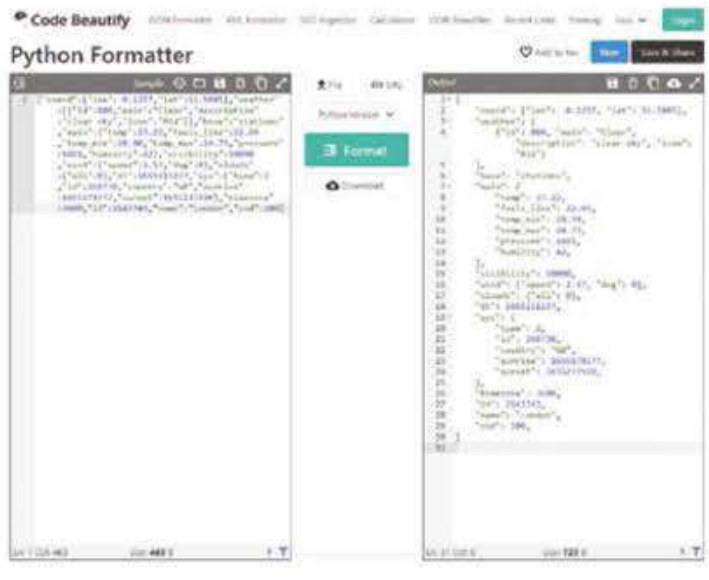
You'll Need

- Raspberry Pi
- Raspberry Pi OS
- Internet connection
- Display-O-Tron HAT (optional) **magpi.cc/displayotron**
- Pibow case (optional)

01 Install Python modules

We'll be using two new Python modules for this project. It's quick and easy to install them. The requests module downloads content from the web using HTTP requests. The feedparser module is used to process RSS feeds, which many news websites use to share headlines, summaries, and links to their stories. Click the Terminal icon on your taskbar to open a terminal window. Then enter the following command at the prompt to download and install the modules:

```
pip install requests feedparser
```



This Python Formatter makes it much easier to understand the JSON structure of the weather data we're using

02 Get your API key

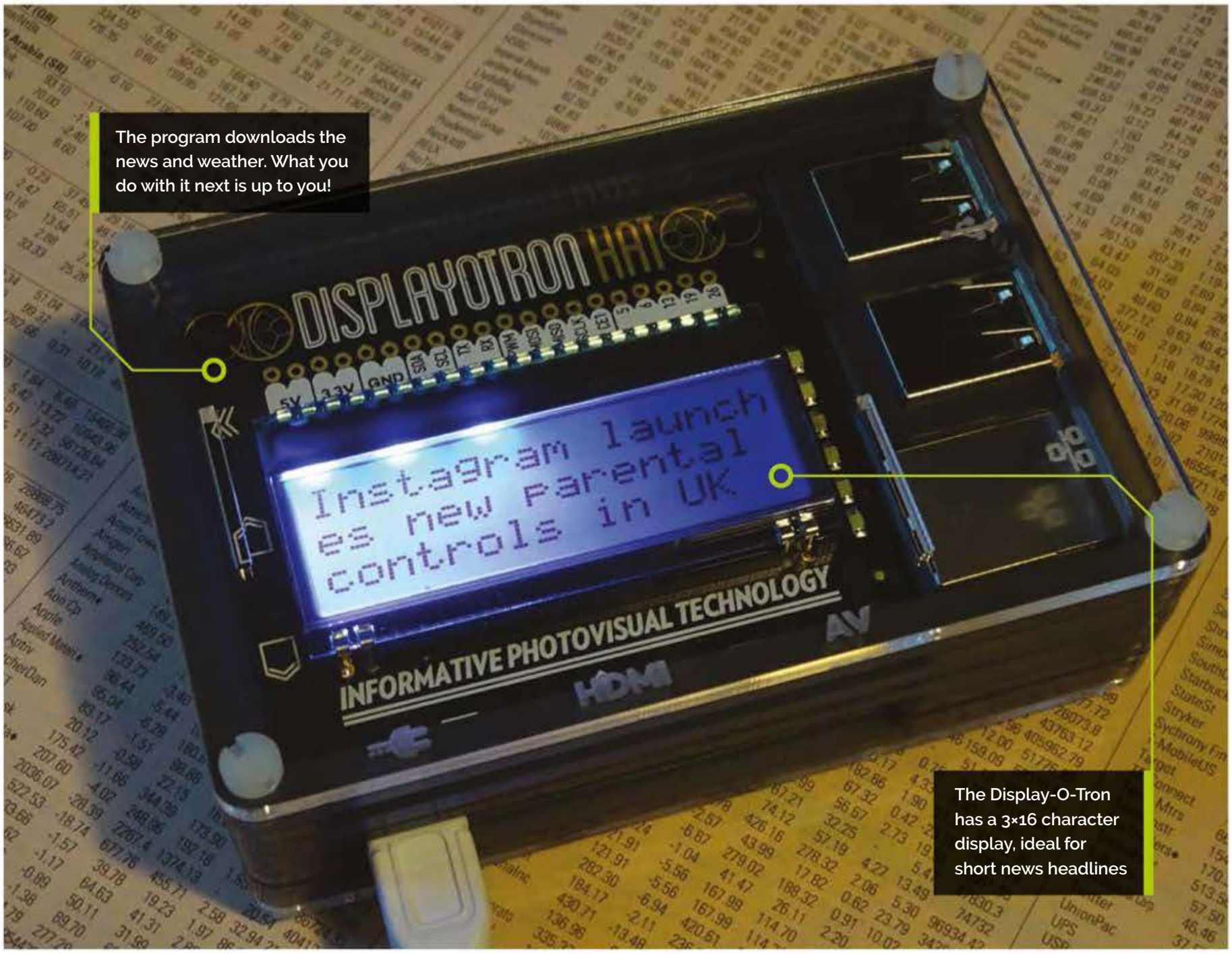
An API (application programming interface) enables applications to talk to each other. In our case, we want our Python program to talk to OpenWeather. The service gives you up to a million requests per month for free, but you need to register for an API key so they can monitor your usage. Visit **openweathermap.org/price** and click 'Get API key' in the free tier. You'll need to register a username, email address, and password, and accept the terms. When you sign into your account, go to your API keys and copy your key.

03 View your weather feed

You'll use a specially formatted URL to fetch the weather. There are lots of different weather forecast options, but we'll keep it simple by looking up the current weather. Let's preview the data feed before we try to use it from Python. In a browser window, enter the following URL.

http://api.openweathermap.org/data/2.5/weather?q=London,uk&units=metric&APPID=YOUR_API_KEY

Change the city name to your own, and add your API key after the equals sign at the end. (If you're not near a city, consult the documentation at **openweathermap.org/current** for tips on using longitude and latitude instead.)



The program downloads the news and weather. What you do with it next is up to you!

The Display-O-Tron has a 3×16 character display, ideal for short news headlines

04 Beautify the output

When you view the feed in your browser, the code isn't formatted for easy reading. It's much easier to understand the incoming data if we reformat it. Copy the code, visit **magpi.cc/beautifier**, and paste the code in the box on the left. Click Format, and the box on the right will show a nicely formatted version of your code. The code is in JSON format, which behaves a bit like a Python dictionary. You use keywords to access the data associated with them. For example, the keyword 'temp' returns a number (measured in Celsius). The keyword 'description' gives me 'clear sky' today.

05 Build the weather report

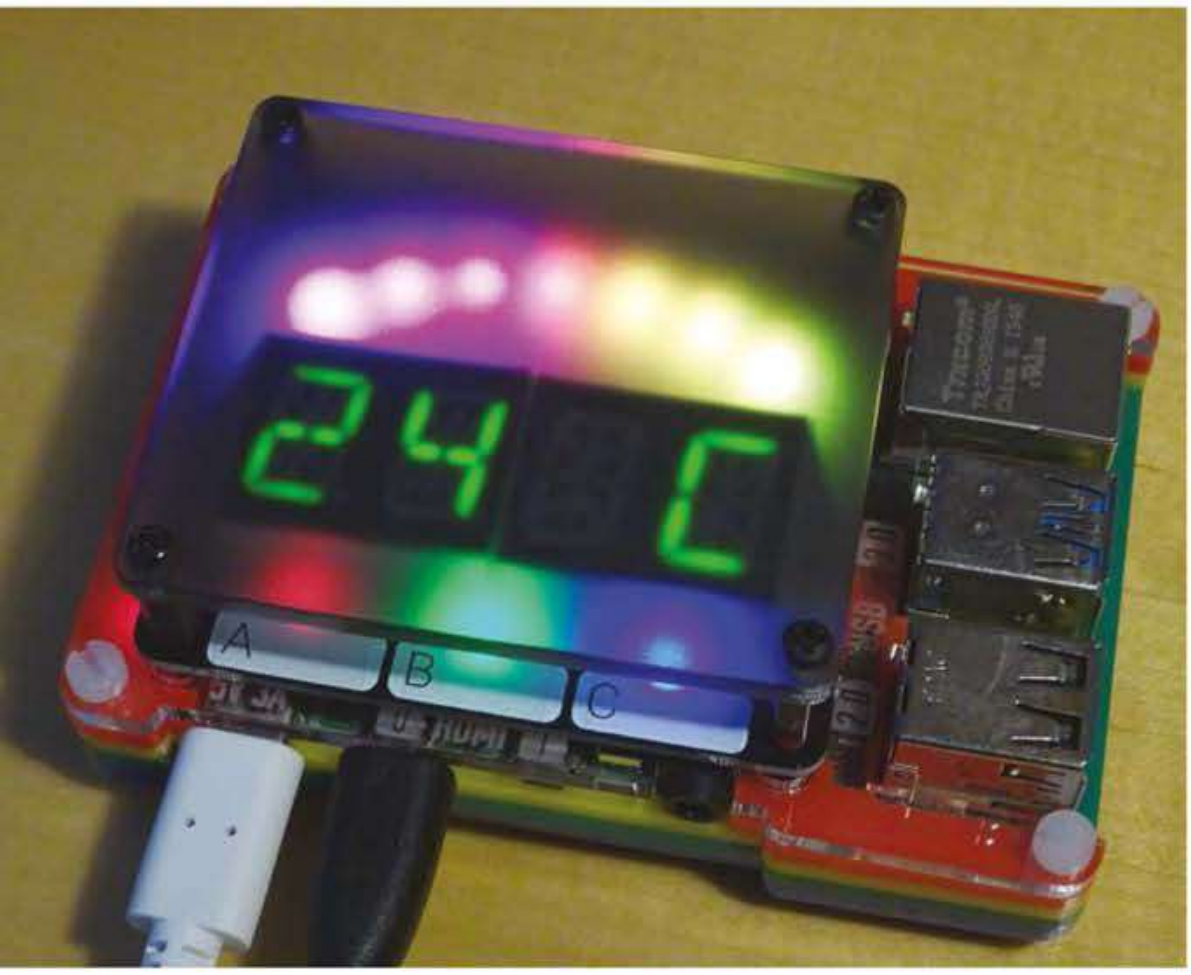
The `get_weather()` function in **rr_newsreader.py** builds a weather report, which is returned as a string. The function also returns the temperature as a number, so you can use it on numeric displays. First, the function uses the requests module to download your weather report. Then it uses the built-in JSON processor in the requests module to extract useful bits of it. Ultimately, Raspberry Radio will read out the weather report, so the report starts with a random phrase to add variety. It's easiest to understand the code if you look at the JSON file (see Step 4) at the same time. The `data` variable stores our full report. We use `data.json().get('weather')` to access the weather attribute of it. The weather section contains an array with one item in it, which we access using `[0]` on line 19. Inside that array, we can find the description attribute. Similarly, we use

> "It's easiest to understand the code if you look at the JSON file"

Top Tip

Extend the weather report

As Step 4 shows, there's more weather data available. Modify the program to report on humidity, wind speed, and what the temperature feels like.



You can use rr_newsreader.py to show the outside temperature on a Rainbow HAT, here shown through a diffuser layer

the attribute 'main' to access a subset of data that includes the 'temp'.

06 Use f-strings for formatting

Line 23 uses a Python f-string to format the weather report. It's an easy way to insert a variable's value into a string. You put an f before the opening quote of the string, and then put the variable name in curly braces inside the string. F-strings are more readable than alternatives, such as percentage formatting and the `format()` function. They both put a placeholder in the string and the associated variable after the string.

news_output.py

Language: **Python**

```
001.  # News display for Displayotron HAT
002.  # From Raspberry Radio project in The MagPi by Sean
      McManus
003.
004.  import rr_newsreader
005.  weather_report, temperature = rr_newsreader.get_weather()
006.  print(weather_report)
007.  date_report = rr_newsreader.get_date()
008.  print(date_report)
009.  news_report = rr_newsreader.get_news()
010.  for line in news_report:
011.      print(line)
```

07 Get an RSS link

Many publications publish RSS feeds, but they're not as prominent on websites as they used to be. You can often uncover them by Google searching for your favourite publication's name plus 'RSS'. The BBC publishes news feeds (**magpi.cc/bbcfeeds**) for topics including technology, health, and entertainment, as well as news feeds dedicated to the different geographies. The Guardian (**magpi.cc/guardianfeeds**) has feeds for a huge number of topics, as diverse as Agatha Christie, the Vietnam War, and Pink Floyd. Raspberry Radio works best if you pick a newsy topic that is fast-moving. We're looking for headlines that make sense in isolation, so avoid feeds promoting feature articles.

08 Get the news

In **rr_newsreader.py**, the `get_news()` function uses the feedparser module to process the RSS feed. Paste the web address of your chosen RSS feed into line 30, or leave the code unchanged for The Guardian's news headlines. The function builds a list called `news_list`. The first list item introduces the news, before the loop extracts the titles from the first three stories in the RSS feed, and adds them (appends them) to the list. The title contains the headline, and it's all we need for our purposes. Each story also has a description (a summary, sometimes quite long), and a link to the main article. To find the URL for the article at index 2, for example, you'd use `rss_feed.entries[2].link`.

09 Get the date

Newsreaders start their broadcast with the date, so the `get_date()` function creates a human-readable date. It uses the datetime module, and its `strftime()` function, which creates a string by extracting parts of the date. You use codes to specify the format you want to use. `%A` gets you

the day name (e.g. Sunday), `%d` extracts the day number, and `%B` gives you the month in full. We're using an f-string again here to build a string that combines those three date elements. There's a list of codes at **strftime.org**. If you used `%I`, `%M`, and `%p`, you would get the time in the format '10 25 AM'.

10 Prepare to import

You might have noticed that **rr_newsreader.py** does not have any output. It defines three functions, but they're not called at any time, and the information they gather isn't displayed. That's because we want to make this code reusable across different projects. This issue, you'll see how to display the news headlines on an LCD screen, but next issue your device will read

> We want to make this code reusable across different projects

them aloud. We can import **rr_newsreader.py** into another Python program, as long as both programs are in the same folder. We used the name **rr_newsreader.py** to reduce the risk of confusion with other newsreader modules.

11 Test the newsreader

The **news_output.py** listing shows how to import **rr_newsreader.py** and access the current date, weather, and news headlines using it. The weather and date are returned as strings. The news is a list, so a loop is used to print each line in turn. You can use this program as a model for collecting the data so you can output it using your favourite HAT.

12 Create your newsreader gadget

The **displayotron_news.py** listing outputs the headlines on a Pimoroni Display-O-Tron. This is just a simple demo: headlines that are too long are shortened to fit the display. You could extend the program so it shows the full headlines and enables you to page through them using the buttons on the HAT. The program uses

You can display the news in the Python shell, as shown here, or use rr_newsreader.py to download it and show it on your favourite HAT

basic animation, displaying each headline one character at a time, and brightening the LEDs when the headline is complete. It makes it feel like the news is coming in right now, and is more visually interesting than just having it pop up on the screen.

Top Tip

Humanise the news

You could add a randomly chosen introduction to the news to increase variety, as we have for the weather.

displayotron_news.py

Language: **Python**

```
001. # News display for Displayotron HAT
002. # From Raspberry Radio project in The MagPi by Sean
     McManus
003.
004. import rr_newsreader
005. import time
006. import dot3k.backlight as backlight
007. import dot3k.lcd as lcd
008.
009. def display_text(text):
010.     backlight.rgb(200, 200, 255)
011.     for i in range(min(len(text), 48)):
012.         substring = text[:i+1]
013.         lcd.clear()
014.         lcd.write(substring)
015.         time.sleep(0.1)
016.     backlight.rgb(255, 255, 255)
017.     time.sleep(3)
018.
019. date_report = rr_newsreader.get_date()
020. display_text(date_report)
021.
022. news_report = rr_newsreader.get_news()
023. for line in news_report:
024.     display_text(line)
```

rr_newsreader.py

DOWNLOAD THE FULL CODE:

magpi.cc/newsreader

> Language: **Python**

```
# rr_newsreader generates news, weather, and date reports
# From Raspberry Radio project in The MagPi by Sean McManus

import feedparser
import requests
import random
from datetime import datetime

def get_weather():
    data = requests.get(
        "http://api.openweathermap.org/data/2.5/weather?q=London,uk&units=metric&APPID=YOUR_API_KEY")
    if data.status_code == 200:
        report = random.choice(["The weather today is ",
                                "We're looking at ",
                                "Today, expect ",
                                "There's going to be ",
                                "Out and about today, you'll see "])
        weather_forecast = data.json().get('weather')
        description = weather_forecast[0].get('description')
        report += description
        main = data.json().get('main')
        temperature = main.get('temp')
        report += f". The temperature is {temperature} Celcius."
        return report, temperature
    else:
        return "There is no weather report today.", False

def get_news():
    news_list = []
    rss_feed = feedparser.parse('https://www.theguardian.com/uk-news/rss')
    news_list.append("Here's the news from The Guardian.")
    for i in range(3):
        news_list.append(rss_feed.entries[i].title)
    return news_list

def get_date():
    date = datetime.today()
    date_text = f"It's {date.strftime('%A')} {date.strftime('%d')} {date.strftime('%B')}."
    return date_text
```

English not your mother tongue?

The MagPi is also available in German!

Subscribe to the German edition of The MagPi and get a Raspberry Pi Pico with headers and a cool welcome box **FOR FREE!**

Use the coupon code 115PicoDE

on www.magpi.de/115

IN THE WORKSHOP: Sublimation printing

By **Ben Everard**

Transfer prints onto mugs and PCBs

Right
The same PCB is used for both left and right wing

A **sublimation printer looks and works a lot like an ordinary inkjet printer.** You load paper into it (although it does have to be proper sublimation paper), plug it into your computer, and press print. So far, so ordinary.

The word sublimation means to turn from a solid into a gas without being a liquid in between. This stage – the sublimation – is what happens after you've printed onto paper. It means you can turn the, now solid, ink into gas and transfer it onto other objects.

When we say other objects – there's a wide range of stuff that you can transfer sublimation prints onto, but not everything. It has to have a coating that will accept the ink. In most cases, this means buying a pre-prepared 'blank', although you can get laminates and varnishes that will let you coat the surface yourself.

Once you have the printed sheet and the object to impress the image on, you just need to stick the paper on with heat-resistant tape (aka Kapton tape), and heat-press it on. This is another limitation on the

You can get heat-presses that can make custom shapes

range of objects that you can transfer images onto – you have to be able to heat-press them.

For flat surfaces, it's pretty easy with a standard heat-press. You can also get cylindrical presses for mugs. Beyond that, it's a bit challenging. For industrial uses, you can get heat-presses that can make custom shapes, but that's probably a little excessive for hobbyist use.

You have to be able to heat the whole lot up to about 200°C, so it's not ridiculously high temperatures. We have heard of people having success using convection ovens and heat-shrink to hold the designs in place, but we've not tried this, and it's probably best not to do it in an oven that's also used for food.

We've been testing out a range of different materials and blanks. Mugs, jigsaws, photo-slates, →

Left
Plain white silkscreen covers the whole of the PCB

Below
A bit of Kapton tape holds the PCBs in place while I heat-press on the image

It's helpful to me if a process is a little tricky

(slates with a flat side that has a layer of sublimation-friendly coating), and fabric.

It's helpful to me if a process is a little tricky or has a few gotchas. This gives me something to write about in the article. The problem is, everything went very smoothly. In fact, the only problem we had in the whole process was a mug heat-press that just kept beeping at us, but reading the manual and working out what button to press sorted that out.

Different materials require slightly different temperatures, and can require radically different times. As far as I can tell, heavy things like ceramics that take a long time to heat up require longer than lighter things, like jigsaw blanks. However, you should be able to get the right time from wherever you get your blanks, if not, you'll just have to experiment for a bit. About 60 seconds at 190°C was the quickest we

Below
I've made mugs for our sister magazines

Left
The butterfly's LEDs can light up any colour

did (jigsaws), and 240 seconds at 205°C was the longest (slate blanks). If you get a sepia tint, leaving your images looking like they are a few hundred years old, then you've probably overheated the ink.

As a final test, we tried sublimating ink onto a PCB, just to see what happens. The initial results were really good. While the ink did not have a great deal of effect on the FR4, it stuck extremely well to silk-screened areas.

This meant that we couldn't just sublimate print onto any PCB we wanted – we had to design specific PCBs covered in silkscreen to take the ink. I've always wanted to make a butterfly PCB. Long-term readers with a very good memory may remember that HackSpace mag, issue 1, featured a butterfly PCB from the now-defunct Boldport Club. In that, they'd used two different silkscreen colours to create the decorative effect but, in mine, I could just print whatever I wanted.

This was just a test, so I wanted something that would look good, but not take too long to design. I settled for a circuit that linked up three through-hole RGB LEDs. The design was one wing, but I could just flip it over to create the other wing.

These came back a week or so later, and it was time to try them out. I didn't have a specific pattern in mind (this was still a test, so I didn't want to spend ages on it before knowing if it'd work), so I printed off a swirling colour image that I found online, stuck a couple of wings to it, crossed my fingers, and started the heat-press.

The result isn't quite perfect. The image doesn't quite have the detail that I've been able to get on some surfaces. I'm not yet sure if that's a result of the inherent properties of the silkscreen, or if I just haven't yet dialled in the settings (the PCBs only arrived a few days before we went to press, so I haven't been able to properly test the process out yet). So far, it seems like the ink is very prone to overheating with PCBs. The best results that I have had are at 180°C for 60 seconds, but it's still early days.

This is definitely a ripe area for experimentation if you're interested in making artistic PCBs. ◻

Part 06

Learn ARM assembly: blink Pico's on-board LED using the Programmable I/O

MAKER

Stephen Smith

Stephen is a retired software developer who has written three books on ARM Assembly Language Programming. He is a member of Sunshine Coast Search and Rescue and enjoys mountain biking, hiking, and running. He is also a member of the Sunshine Coast Writers and Editors Society (**scwes.ca**).

magpi.cc/stephensmith

Learn how to use the Programmable I/O (PIO) processors on the Raspberry Pi Pico in their native assembly language to blink the on-board LED

Directly controlling hardware devices with the ARM CPU, like we did in the previous tutorial (see *The MagPi* issue #120, magpi.cc/120), uses a lot of processing power that you might need for higher-level tasks. The Raspberry Pi Pico contains eight PIO processors that you can use to offload I/O related processing. In this tutorial, we will blink the Raspberry Pi Pico's on-board LED. The program downloads a small assembly language program to one of the PIO processors which will then do all the work of blinking the LED, letting the main ARM CPU proceed to perform other tasks.

The PIO processors operate on their own assembly language, which is quite different from ARM assembly language. This assembly language only contains nine instructions, and the maximum program size is 32 instructions, but even with these limitations, PIO can perform quite powerful I/O operations. There are five registers: two general-purpose registers, two shift registers for data transfer, and the program counter.

01 Create the program

Create a folder named **tutorial6** in the **pico** folder that was created in your home folder by the Raspberry Pi Pico SDK's setup script. The source code for this tutorial is in **blink.c**, **blink.pio** and **CMakeLists.txt**. Copy the file **pico_sdk_import.cmake** from the SDK's external folder to the **tutorial6** folder. In this folder create a new folder called **build**. The **tutorial6** folder should now look like:

```
pi@raspberrypi:~/pico/tutorial6 $ ls -l
total 20
-rw-r--r-- 1 pi pi  972 Jun 25 11:22 blink.c
-rw-r--r-- 1 pi pi  925 Jun 25 11:17 blink.
pio
drwxr-xr-x 7 pi pi 4096 Jun 25 11:22 build
-rw-r--r-- 1 pi pi  698 Nov  5  2021
CMakeLists.txt
-rw-r--r-- 1 pi pi 2763 Apr  9 09:15 pico_
sdk_import.cmake
```

02 Build the program

These steps are identical to those in tutorial 5. The difference is that the **CMakeLists.txt** file adds includes a **pio** file containing PIO assembly language rather than the usual **.s** ARM assembly language file. Open a Terminal window and `cd` to the **tutorial6** folder.

```
cd pico/tutorial6/build
```

Run `cmake` with the option to perform a debug build. Placing this command in a script file in the **$HOME/bin** folder with a short filename can be a real time-saver.

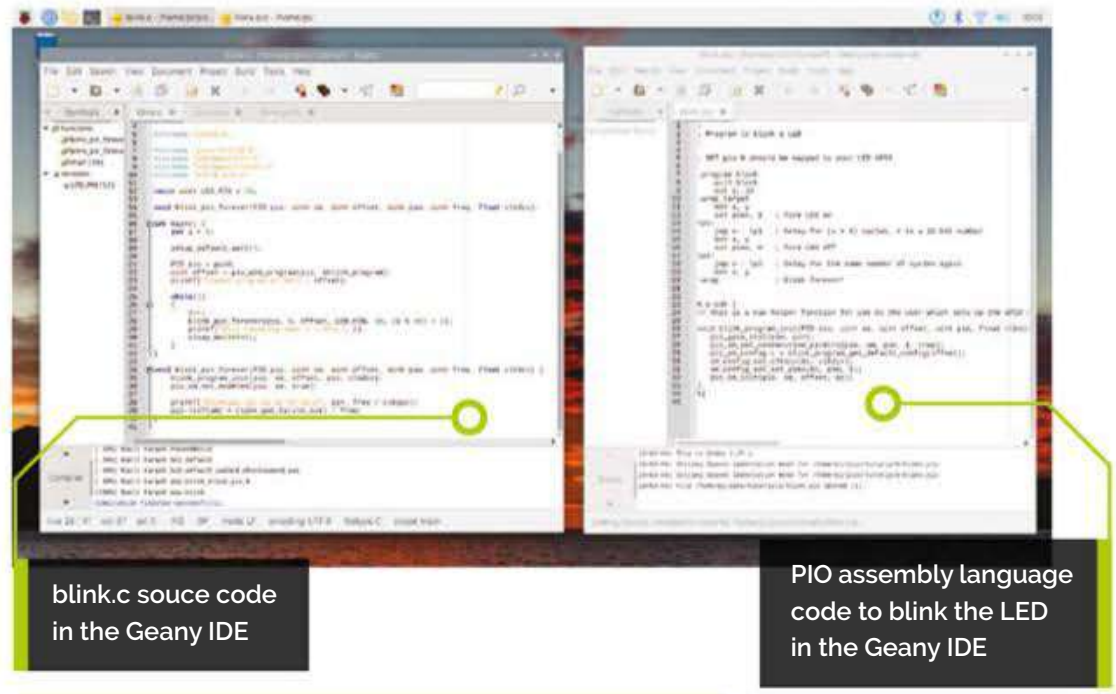
```
cmake -DCMAKE_BUILD_TYPE=Debug ..
```

The `cmake` command doesn't build the program – instead, it creates a makefile used to compile by running `make`:

```
make
```

The **build** folder should now look like the example below:

```
 pi@raspberrypi:~/pico/tutorial6/build $ ls
-l
 total 1192
 -rw-r--r-- 1 pi pi   1856 Jun 25 11:19
blink.pio.h
 -rw-r--r-- 1 pi pi  18815 Jun 15 13:30
CMakeCache.txt
 drwxr-xr-x 7 pi pi   4096 Jun 25 11:22
CMakeFiles
 -rw-r--r-- 1 pi pi   1670 Jun 15 13:30
cmake_install.cmake
 drwxr-xr-x 6 pi pi   4096 Jun 15 13:30
elf2uf2
 drwxr-xr-x 3 pi pi   4096 Jun 15 13:30
generated
 -rw-r--r-- 1 pi pi  77139 Jun 15 13:30
Makefile
 drwxr-xr-x 6 pi pi   4096 Jun 15 13:30 pico-
sdk
 drwxr-xr-x 5 pi pi   4096 Jun 15 13:31
pioasm
 -rwxr-xr-x 1 pi pi  22652 Jun 25 11:22 pio_
blink.bin
 -rw-r--r-- 1 pi pi 364370 Jun 25 11:22 pio_
blink.dis
 -rwxr-xr-x 1 pi pi 351404 Jun 25 11:22 pio_
blink.elf
 -rw-r--r-- 1 pi pi 234360 Jun 25 11:22 pio_
```



blink.c souce code in the Geany IDE

PIO assembly language code to blink the LED in the Geany IDE

```
blink.elf.map
 -rw-r--r-- 1 pi pi  63776 Jun 25 11:22 pio_
blink.hex
 -rw-r--r-- 1 pi pi  45568 Jun 25 11:22 pio_
blink.uf2
```

03 Run the program

Power on Raspberry Pi Pico by plugging the USB cable into the power connector while holding down the BOOTSEL button. When the file explorer window appears, open it, and copy the file **pio_blink.uf2** to the Pico. Your Pico reboots, then the program runs. The program prints out information on how fast the LED is blinking and which iteration of the loop it is on. Open a serial port program to see this output.

> PIO can perform quite powerful I/O operations

```
minicom -b 115200 -o -D /dev/serial0
```

The `minicom` command displays data being sent from the Pico to Raspberry Pi (**Figure 1**).

04 About compiling the PIO module

In previous tutorials, the GNU Assembler compiled our assembly language files into object modules that were linked into the program's executable file, which would be run directly by the operating system. The PIO code needs to be separately downloaded to the PIO program memory via Raspberry Pi Pico's SDK. The PIO assembler

You'll Need

- Raspberry Pi
- Raspberry Pi Pico
- Serial and debug connector wires
- Raspberry Pi OS 32-bit
- Raspberry Pi Pico SDK

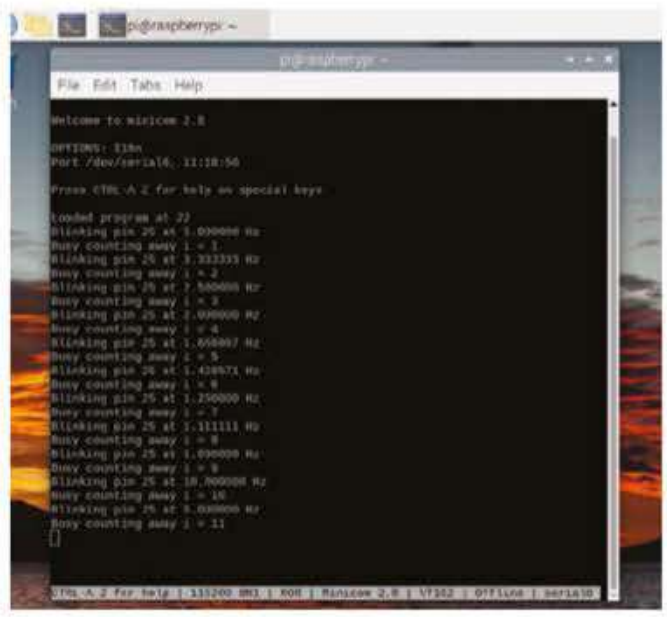


Figure 1 Running the program and observing the output in minicom

compiles **blink.pio** into **blink.pio.h**, which is a standard C header file. This file contains an array of 16-bit numbers that are the compiled values of the program:

```
  static const uint16_t blink_program_
instructions[] = {
      0x80a0, //  0: pull   block
      0x6040, //  1: out    y, 32
              //     .wrap_target
      0xa022, //  2: mov    x, y
      0xe001, //  3: set    pins, 1
      0x0044, //  4: jmp    x--, 4
      0xa022, //  5: mov    x, y
      0xe000, //  6: set    pins, 0
      0x0047, //  7: jmp    x--, 7
      0xa022, //  8: mov    x, y
              //     .wrap
};
```

The header file includes several helper functions to set various configuration parameters. The program is downloaded via the `pio_add_program` SDK function called from the `main()` function in **blink.c**.

CMakeLists.txt

Language: **CMake**

```
001.  cmake_minimum_required(VERSION 3.13)
002.
003.  include(pico_sdk_import.cmake)
004.  project(test_project C CXX ASM)
005.
006.  set(CMAKE_C_STANDARD 11)
007.  set(CMAKE_CXX_STANDARD 17)
008.
009.  pico_sdk_init()
010.
011.  add_executable(pio_blink)
012.
013.  # by default the header is generated into the build dir
014.  pico_generate_pio_header(
      pio_blink ${CMAKE_CURRENT_LIST_DIR}/blink.pio)
015.  # however, alternatively you can choose to generate it
      somewhere else (in this case in the source tree for check
      in)
016.  #pico_generate_pio_header(pio_blink ${CMAKE_CURRENT_LIST_
      DIR}/blink.pio OUTPUT_DIR ${CMAKE_CURRENT_LIST_DIR})
017.
018.  target_sources(pio_blink PRIVATE blink.c)
019.
020.  target_link_libraries(pio_blink PRIVATE pico_stdlib
      hardware_pio)
021.  pico_add_extra_outputs(pio_blink)
```

05 How the PIO assembly language works

In all the previous tutorials, we studied how the program works by single-stepping through it in the gdb debugger. As of this writing, there isn't an open-source debugger for the PIO assembly language. This tutorial program contains just eight instructions. To get a feel for PIO programming, let's walk through these instructions.

The first thing the program does is receive the count for a delay loop from the main program running on the ARM CPU. It takes two instructions to do this:

- `pull block` – moves 32 bits of data from the transmit FIFO to the PIO's Output Shift Register (OSR). The FIFO is transmit, since data is transmitted from the ARM CPU.
- `out y, 32` – transfers 32 bits from the OSR to the general-purpose Y register.

Not all operations on the PIO are performed via instructions – several are set by configuration. For instance, the statements `.wrap_target` and `.wrap` are for configuration. If the program counter (PC) passes instruction 31, then it wraps back to instruction 0. However, there are control registers that let you configure when the PC will wrap and which instruction it will wrap to. This allows you to have one infinite loop without using one of the valuable 32 assembly instructions to perform the loop.

The `mov` instruction is straightforward: here, the delay loop count is moved from the Y to the X register, ready to be counted down.

The `set` instruction is used to set the configured GPIO pins either high or low. First, it is set high

to turn the LED on. The actual pin affected is controlled by a configuration register.

The `jmp` instruction does several things. It is the only PIO instruction that performs arithmetic, and the only arithmetic it can perform is subtracting one. It provides conditional logic that will jump so long as the register, in this case `X`, is non-zero. Once `X` is zero, it falls through. The conditional logic can be reversed by placing a `!` in front of the register. The purpose of the `jmp` statements in this program is to provide a delay so the LED flashes at a rate a human can perceive rather than at the speed of the Raspberry Pi Pico's 125MHz clock.

> "Normally, each instruction takes one clock cycle"

These instructions then repeat, turning the LED off and performing the same delay loop to result in an even blinking.

06 Varying the timing

The primary design goal of the PIO processors is to handle the lower-level details of various communications protocols, which involves switching GPIO pins between high and low at high speeds subject to precise timing. The PIO offers quite a few features to assist in this. For instance, each instruction can have an extra parameter that provides up to five bits telling it how many clock cycles to take. Normally, each instruction takes one clock cycle. But, for instance:

```
mov x,y [31]
```

...will take 32 clock cycles to execute: one to perform the move, then 31 extra cycles for a total of 32.

This tutorial uses the PIO's clock divider to vary the speed at which the LED blinks. The clock divider has a 1/256 precision, so its value is provided as a floating-point number to the SDK, then the SDK converts that into the value required by the PIO control register. We set the clock divider in the `blink_program_init` routine in **blink.pio** via:

```
sm_config_set_clkdiv(&c, clkdiv);
```

To change the value, the PIO is reconfigured every ten seconds from the main C file.

blink.pio

Language: **PIO Assembly Language**

```
001.  ;
002.  ; Program to blink a LED
003.  ;
004.
005.  ; SET pin 0 should be mapped to your LED GPIO
006.
007.  .program blink
008.      pull block
009.      out y, 32
010.  .wrap_target
011.      mov x, y
012.      set pins, 1   ; Turn LED on
013.  lp1:
014.      jmp x-- lp1   ; Delay for (x + 1) cycles, x is a 32
      bit number
015.      mov x, y
016.      set pins, 0   ; Turn LED off
017.  lp2:
018.      jmp x-- lp2   ; Delay for the same number of cycles
      again
019.      mov x, y
020.  .wrap             ; Blink forever!
021.
022.
023.  % c-sdk {
024.  // this is a raw helper function for use by the user
      which sets up the GPIO output, and configures the SM to
      output on a particular pin
025.
026.  void blink_program_init(
      PIO pio, uint sm, uint offset, uint pin, float clkdiv) {
027.     pio_gpio_init(pio, pin);
028.     pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true);
029.     pio_sm_config c =
      blink_program_get_default_config(offset);
030.     sm_config_set_clkdiv(&c, clkdiv);
031.     sm_config_set_set_pins(&c, pin, 1);
032.     pio_sm_init(pio, sm, offset, &c);
033.  }
034.  %}
```

07 The main C program

The main C program initialises the PIO and lets it do its job. The program can start the LED blinking and then go on to do other processing without having to worry about the LED ever again. This is the goal of offloading low-level I/O

operations to the PIO, freeing up the ARM CPUs to perform other useful work. In this case, after initialising the PIO, the C program executes a loop where every ten seconds it re-initialises the PIO with a different timing value based on the index it counts.

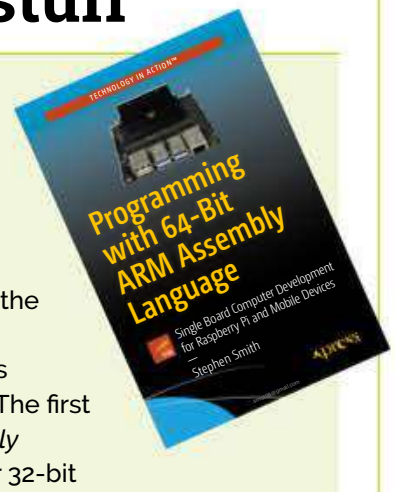
Raspberry Pi's SDK examples contain 19 PIO sample programs. The best way to write a PIO program is to use one of these as a starting point, if there isn't already one that meets your needs. Then refer to the 'Raspberry Pi Pico C/C++SDK' and 'RP2040 Datasheet' manuals for the various SDK functions that control the PIO, and for details on PIO assembly language.

08 Modify the program

Congratulations! Welcome to the world of PIO programming for Raspberry Pi Pico (and any other board using Raspberry's RP2040 chip). Without debugging or print statements, developing PIO programs can be tricky. The best way is to start simple, such as taking a working program like this one and modifying it to change its functionality. Then, as long as you take small steps, when something breaks, you will know what did it and you can go back to a working state. Try experimenting with the timing of the LED pattern, perhaps have it do a longer flash followed by a shorter flash. Can you think of a way to expand this to transmit Morse code?

Stephen's stuff

He's written three books on Assembly Language Programming. The most recent one is *RP2040 Assembly Language Programming* for the Raspberry Pi Pico, which is the place to go for a deeper understanding of the topics touched on in this tutorial. The first one is *Raspberry Pi Assembly Language Programming* for 32-bit ARM code, and the second is *Programming with 64-Bit ARM Assembly Language*.



blink.c

DOWNLOAD THE FULL CODE:
magpi.cc/learnassembly6

Language: **C**

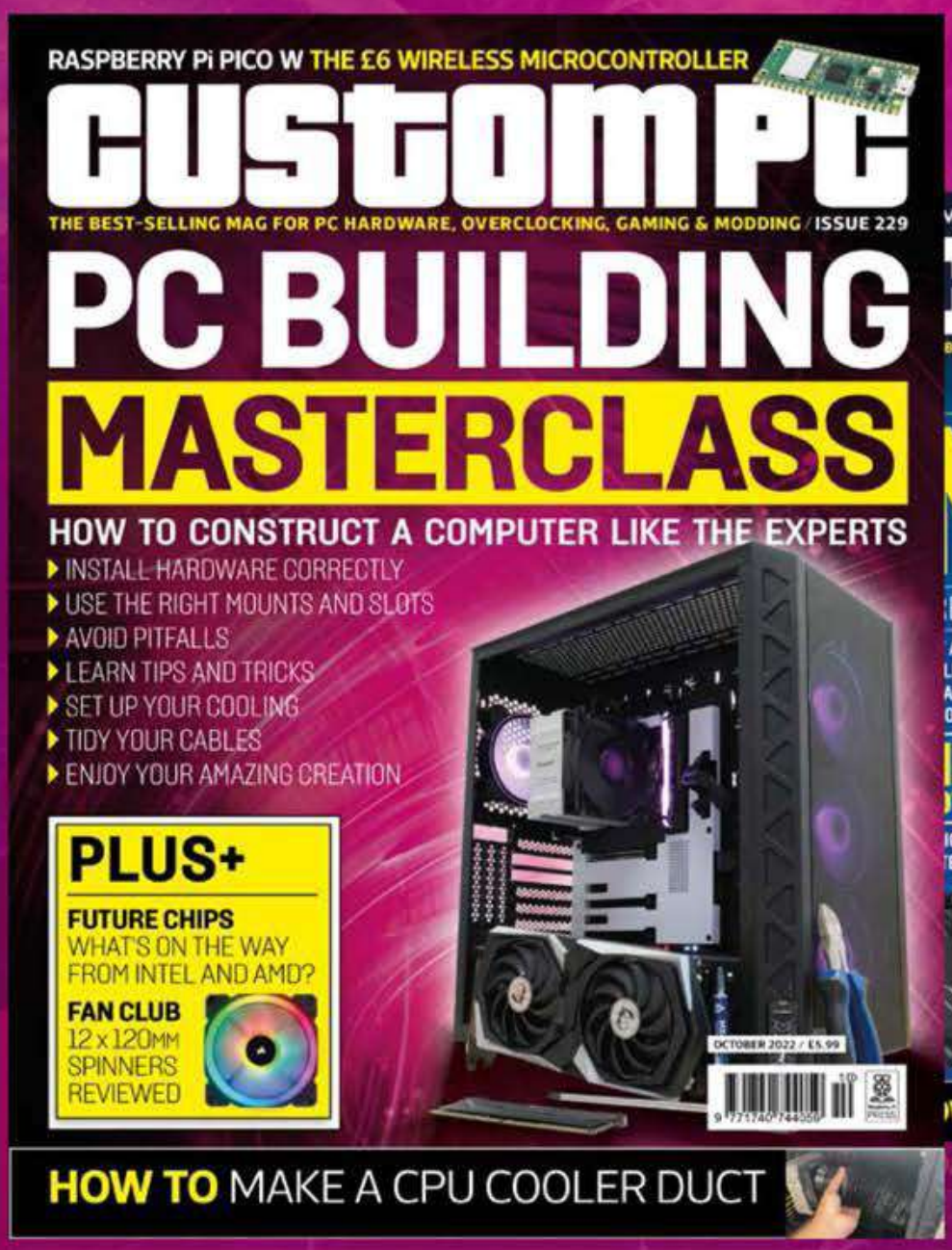
```
/**
 * C Program to set the PIO in motion blinking the LEDs
 */

#include <stdio.h>

#include "pico/stdlib.h"
#include "hardware/pio.h"
#include "hardware/clocks.h"
#include "blink.pio.h"

const uint LED_PIN = 25;

void blink_pin_forever(PIO pio, uint sm, uint offset,
uint pin, uint freq, float clkdiv);

int main() {
    int i = 0;

    setup_default_uart();

    PIO pio = pio0;
    uint offset = pio_add_program(pio, &blink_program);
    printf("Loaded program at %d\n", offset);

    while(1)
    {
        i++;
        blink_pin_forever(pio, 0, offset, LED_PIN, 10, (
i % 10) + 1);
        printf("Busy counting away i = %d\n", i);
        sleep_ms(10000);
    }
}

void blink_pin_forever(PIO pio, uint sm, uint offset,
uint pin, uint freq, float clkdiv) {
    blink_program_init(pio, sm, offset, pin, clkdiv);
    pio_sm_set_enabled(pio, sm, true);

    printf("Blinking pin %d at %f Hz\n", pin,
freq / clkdiv);
    pio->txf[sm] = clock_get_hz(clk_sys) / freq;
}
```

CUSTOM PC

THE BEST-SELLING MAG FOR PC HARDWARE, OVERCLOCKING, GAMING & MODDING

THE MAGAZINE FOR

PC HARDWARE ENTHUSIASTS



ISSUE 229 OUT NOW

VISIT **CUSTOMPC.CO.UK** TO LEARN MORE

Part 02

Build a robot: add sensors to the chassis

Last month, we started our build of the CamJam Robotics EduKit. Now we have a roving robot, it's time to add some smarts!

MAKER

PJ Evans

PJ is a writer, software engineer and tinkerer. His robots bring all the geeks to the yard.

twitter.com/mrpjevans

If you followed last month's tutorial, you should now have a working robot that you can control with Python. Hopefully, you've played with the code and had the little 'bot' zooming around the place. Now it's time to add some sensors, so our new pal can start to sense the world around it. With the ultrasonic and light sensors included with the CamJam EduKit #3, we can add some autonomous capabilities and make our robot a little smarter. Finally, we can look at what you can do to improve the robot even more with custom chassis and additional sensors.

You'll Need

- CamJam Edukit #3 - Robotics **magpi.cc/edukit3**
- Printer
- Roll of paper (optional)

01 Get sensitive

Included with your CamJam kit is a light sensor. It works by sending out infrared light (that we can't see) and detecting how much of it bounces back to a sensor. If we point it at the ground and measure the sensor's output, we can easily tell when the robot passes over a line. The key to success is high contrast, so a jet black line on a white surface is perfect. We're going to mount the sensor on the front of the robot, point down, so we can detect a line.

02 A little light wiring

To wire up the light sensor, we're going to use the breadboard (the small block with lots of holes). Holding it with the longer edge horizontal, each column of holes are connected together, with a gap in the centre. Breadboards allow us

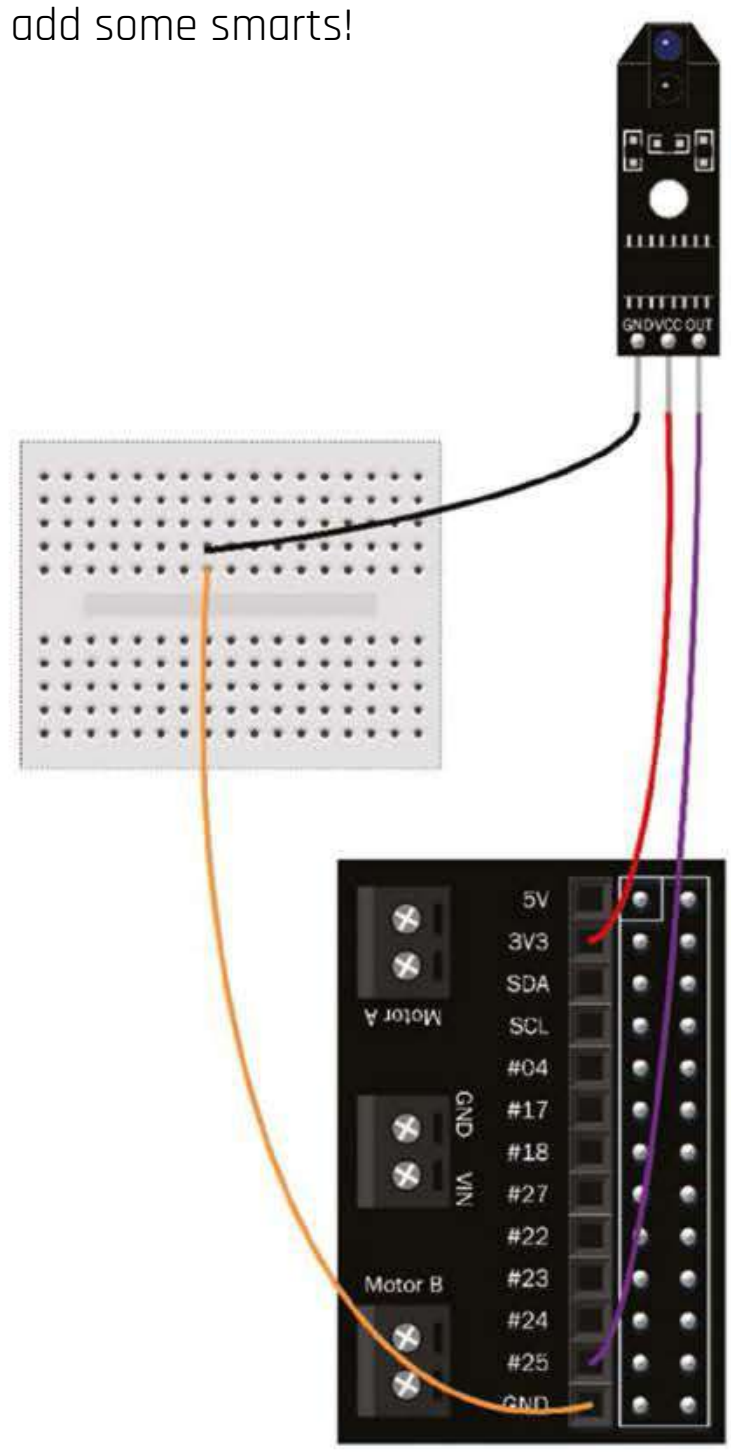
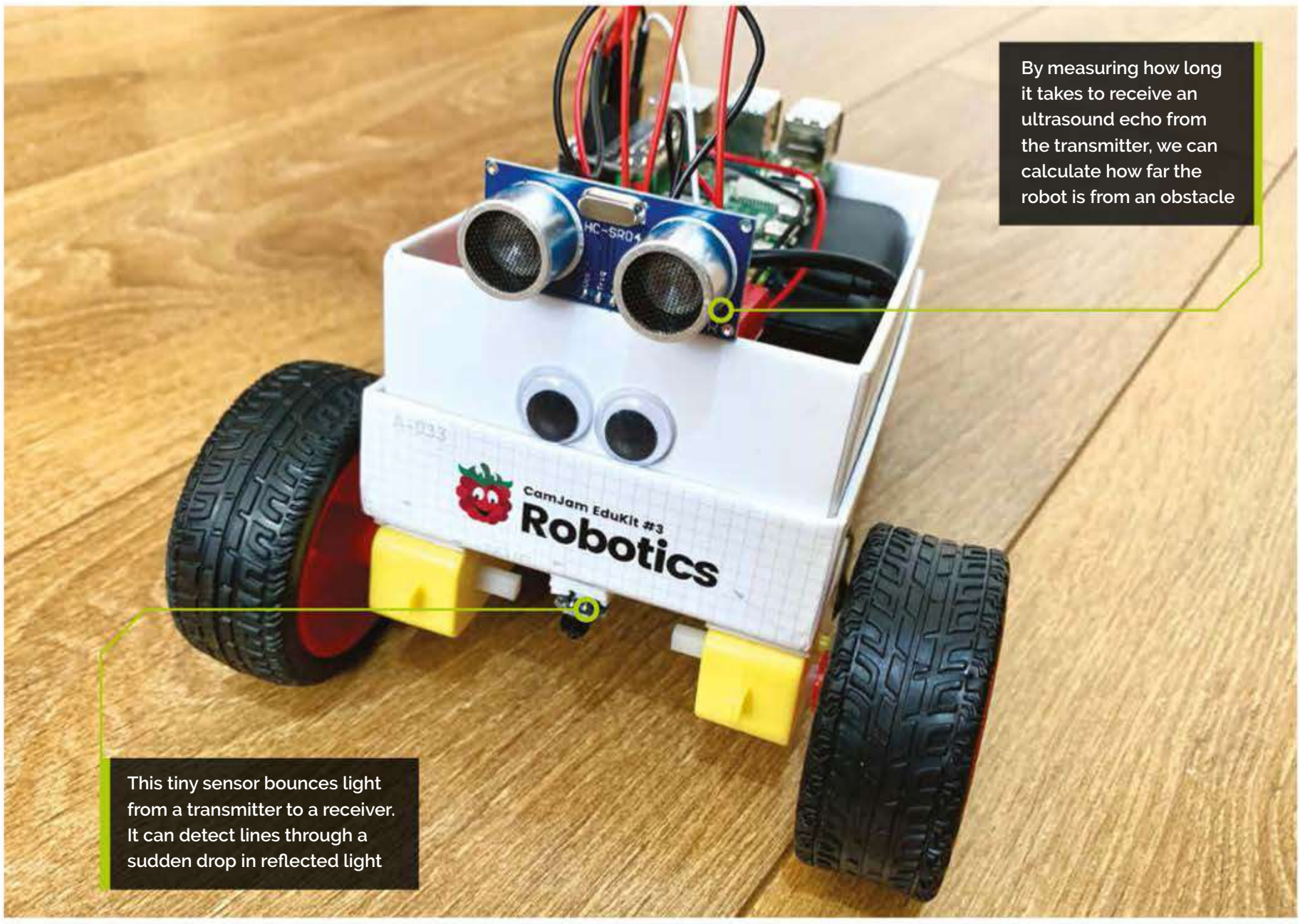


Figure 1 Wire up the line sensor to the HAT. Use the breadboard to create a 'ground rail'



By measuring how long it takes to receive an ultrasound echo from the transmitter, we can calculate how far the robot is from an obstacle

This tiny sensor bounces light from a transmitter to a receiver. It can detect lines through a sudden drop in reflected light

to connect circuits together without soldering, so we can quickly prototype circuits and correct mistakes. You'll need three wires: two plug-to-socket, and one plug-to-plug. Wire everything together as shown in the diagram, checking and double-checking. There are three connectors on the light sensor for power, grounding, and data. These all need to match up with the connector on the robot HAT connected to your Raspberry Pi.

03 Mount the sensor

The sensor needs to be in a sensible place on the box chassis, and that would be in the centre at the front on the base. However, this is also the highest point of the body, and the further away the sensor is from the ground, the less accurate it will become as ambient light leaks into the sensor. Start by making a hole off-centre towards the front of the chassis and feeding the three wires through it. Connect the wires to the sensor as shown in **Figure 1**, then mount the sensor to the body with sticky pads. You may find a couple of LEGO bricks will sufficiently lower the sensor to the ground.

04 Testing time

The sensor will not work without a little code to help it on its way. Enter the code from the **test_line.py** listing, overleaf (or download it from the GitHub repo: **magpi.cc/testlinepy**), and save it in your home directory as **test_line.py**. Now run the code:

```
python3 test_line.py
```

Don't worry, your robot won't move at this point. What we want to do is check the sensor is working correctly. Using a sheet of paper with a thick black line through it (**magpi.cc/testlinepdf**), hold your robot carefully and pass the paper under the sensor. If all is working well, you'll see messages on-screen that the line has been detected.

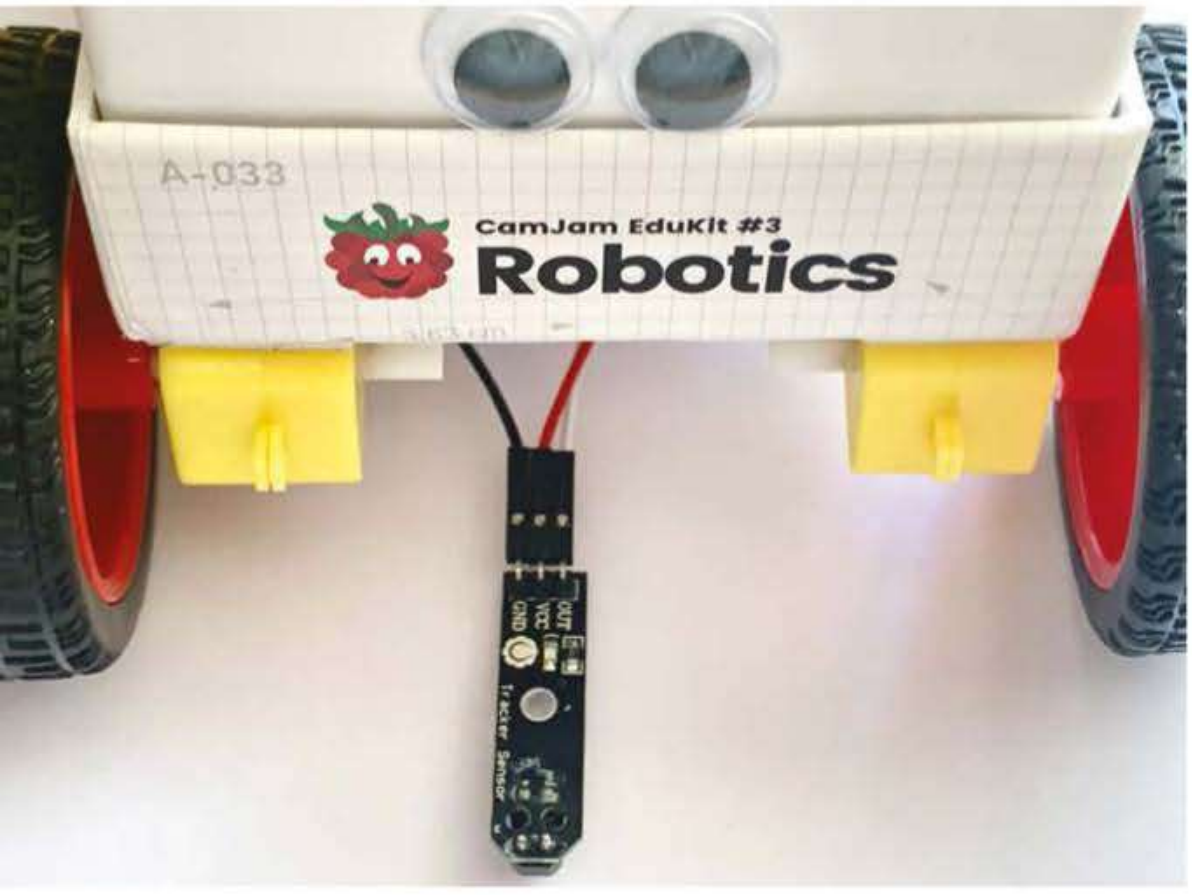
05 Follow that line

Now our little pal can detect a line, we can make them follow it too! By combining code to drive the robot forward and steer it left and right, we can make corrections as we go. This listing is a

Top Tip

Safety First

Whenever connecting wires to your Raspberry Pi computer, ensure it is completely powered-down. A mistake when the computer is on can cause permanent damage.



▲ Here is the wiring on the line sensor. The lower the sensor can be mounted to the ground, the more accurate the result will be

little longer, so download it to your home directory from **magpi.cc/linefollowerpy** and try it out. Place your robot at the start of the line and then run:

```
python3 line_follower.py
```

Hopefully, your roving friend will scoot along the length of the page. Do you notice how it's slower and more controlled? We're using pulse-width modulation (PWM) to slow the motors down. You can play with the setting by changing the `leftmotorspeed` and `rightmotorspeed` variables.

06 Make a course, of course

OK, so our new friend can follow a line. How about an entire course? If you've got access to a big roll of white paper, try mapping out a course for the robot to follow. If not, you could stick pieces of A4 paper together. Make it as big as you can, without any tight corners to which the robot may not be able to respond. Use a pen such as a Sharpie to create the line to follow, and make it nice and thick, like the one on the printout. Now watch as your newly smarter robot follows the line in circles.

07 Looking into the distance

If we want our robot to be able to move around a room on its own, there's a significant problem: walls. Our final modification is to add a distance sensor to the robot, so it can take avoiding action when it gets close to an obstacle. The sensor works by transmitting an ultrasonic pulse and detecting when it is returned. With a bit of maths, we can use the time taken ('time of flight') to calculate how far away the obstacle is from the robot. The wiring is a little more complicated for this as the sensor needs 5V to work, but must only return a 3.3V signal to avoid damaging your Raspberry Pi 4.

08 Wiring up for safe voltage

Study the **Figure 2** wiring diagram carefully. Mount the sensor to the breadboard along the long edge, so each connector has its own column (or 'rail'). Move the two existing ground connectors for the line sensor so they are on the same rail as the GND pin for the distance sensor. Connect TRIG to #17 on the CamJam HAT. Finally create a 'voltage divider' to reduce the return voltage to 3.3V. Do this by adding the supplied 470Ω resistor to bridge the GND rail to any spare rail. Now add the 330Ω resistor to bridge that spare rail to ECHO. Finally, link the the spare rail to #18 on the HAT.

Top Tip

Colourful resistors

Resistors are identified by coloured stripes on their body and can be used either way around. The 470Ω resistor is Yellow, Violet, Black, Black, Brown; the 330Ω is Orange, Orange, Black, Black, Brown.

test_line.py

DOWNLOAD THE FULL CODE: magpi.cc/testlinepy

▶ Language: **Python 3**

```
001. import time
002. from gpiozero import LineSensor
003.
004. # Set variables for the GPIO pins
005. pinLineFollower = 25
006. sensor = LineSensor(pinLineFollower)
007.
008. def lineseen():
009.     print("Line seen")
010. def linenotseen():
011.     print("No line seen")
012.
013. # Tell the program what to do with a
     line is (un)seen
014. sensor.when_line = lineseen
015. sensor.when_no_line = linenotseen
016.
017. # Repeat the next indented block
     forever
018. while True:
019.     time.sleep(10)
```

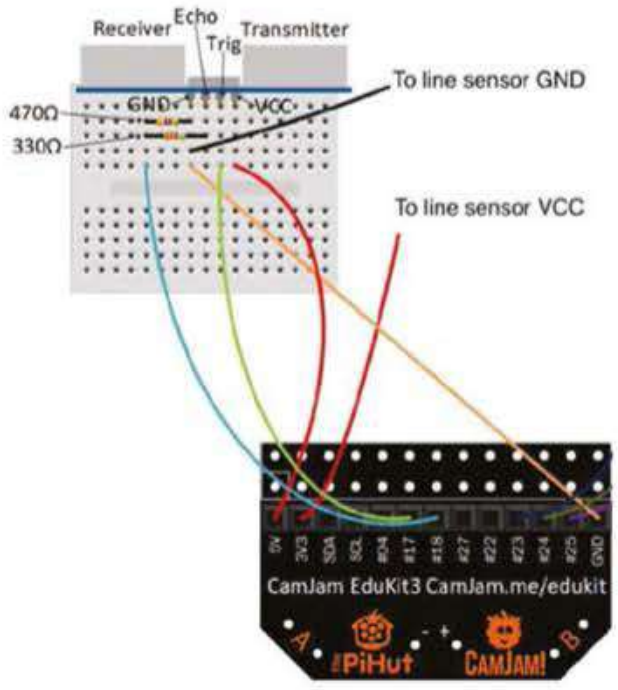


Figure 2 Build this circuit to get readings from the distance sensor. Always triple-check everything before powering-up!

09 Facing forward

To get an accurate reading, the ultrasonic sensor needs to be mounted facing forward in the centre. You may have to get a little creative to find the best way to attach the breadboard so it fits. We used a bit of double-sided sticky tape on the breadboard to hold in place, so the sensor sat over the edge of the box. A small cardboard or plastic box for the board to sit on would also work well. Another option is to remove the sensor from the breadboard altogether and use four jumper wires to reconnect it, allowing the breadboard to sit on the base.

10 Testing from a distance

Let's create another test file. In your home directory, create a new file called **test_distance.py** and add the code from the listing here (or download it from **magpi.cc/testdistancepy**). As before, run this code from the command line:

```
python3 test_distance.py
```

Watch the output from the screen and move your hand towards the sensor. If everything is working, you'll see measurements of the distance from your hand to the robot. This is calculated by taking the output from the sensor (the elapsed echo time in seconds), multiplying it by the speed of sound (34,326 cm per second) and then halving, as it has made an outward and return journey.

11 You own autonomous robot

Congratulations! Your robot build is now complete. Let's combine all the parts of the robot in one last Python program. It's a bit long, so you can download it here: **magpi.cc/avoidancepy**. The code will move the robot forward until it detects an upcoming obstacle. It will then back off and turn right. It will then advance forward. If the obstacle is still in place (or a new one is found), it will repeat the process until the obstacle is cleared. These simple instructions will result in a robot that will happily trundle around the room until you stop it or the batteries wear out!

12 Make it your own

You are the proud custodian of a line-following, obstacle-detecting robot. The good news is, that's not the end of your, or your robot's, journey. Now you have the basic building blocks of code, try and make your robot do more. Could you add dynamic speed control based on distance from an obstacle? Are there other sensors you could add? How about 3D-printing Daniel Bull's custom-designed chassis (**magpi.cc/robotchassis**)? Or, use this project as a starting point and design and build your own robot. Try replacing the CamJam HAT with a motor controller board and upgrading to four-wheel drive. Whatever you decide, have fun.

Many thanks to Mike Horne and Tim Richardson of CamJam for their help sheets and code that informed this tutorial.

test_distance.py

DOWNLOAD THE FULL CODE: magpi.cc/testdistancepy

Language: **Python 3**

```
001. import time
002. from gpiozero import DistanceSensor
003.
004. # Define GPIO pins to use on the Pi
005. pintrigger = 17
006. pinecho = 18
007. sensor = DistanceSensor(echo=pinecho,
     trigger=pintrigger)
008.
009. # Check the distance every half a second
010. while True:
011.     print("Distance: %.1f cm" % (
     sensor.distance * 100))
012.     time.sleep(0.5)
```

SMART AND SPOOKY HALLOWEEN PARTY!

Use Raspberry Pi and Pico to power up your petrifying party this Halloween with Ghost Host **Rob Zwetsloot**

Halloween parties are great. When else do you have an excuse to eat unusual amounts of sweets while watching a marathon of *The Simpsons* episodes, and maybe partaking of a beverage that is a) smoking and b) served out of a cauldron?

This year, let's make this a Halloween party to remember by performing some mad science with a Raspberry Pi or Raspberry Pi Pico to make everything a little more… eerie.

You won't need strikes of lightning for this, just some long USB leads and/or batteries. Let's get spooky.

ATROCIOUS ALTERNATE ARTICLES

Spook up your Raspberry Pi

magpi.cc/110

Make your existing Raspberry Pi projects seasonally spooky with this frightening feature.

Haunted House Hack

magpi.cc/98

A gruesome guide for making your home horrifying this Halloween.

10 Spooktastic Halloween Costume Projects

magpi.cc/86

Dreadful dressing-up with this collection of chilling cosplays.

HOUSE OF HORRORS

Gruesome garlands to meet your ghastly guests

LURID LIGHTS

Holiday light show

Luke Dutton

While, yes, technically these are Christmas-themed lights, you just need to switch the light colours and music to make it truly Halloween-themed.

It uses LightShow Pi, which is a popular library for syncing up music to programmable LEDs to make your whole setup seem in sync. We imagine it would look quite scary with the *Monster Mash* playing.

magpi.cc/holidaylights

PUMPKINS 'N' PROPS

Poplawski Holiday Frights

Chris Poplawski

A huge project full of outdoor Halloween decorations controlled online by people around the world might sound chaotic – and it is – however that's also a big part of the Halloween spirit, surely?

What started off as Christmas-themed lights was turned spooky for October. One credit to activate one thing was around 10¢ (8p). However, you don't have to make yours pay-to-play.

magpi.cc/poplawski

DIY Interactive Performing Pumpkins

DIY Machines

These 3D-printed pumpkins use projections to serenade you with some scary songs, and people passing by can even select different moods for the melodies. While this may sound like a bit of a weather hazard for the projector, this particular project allows you to keep it inside, safe from the elements.

There's a very detailed tutorial on the DIY Machines site, including wiring diagrams and code, etc. Pretty much any projector will do, with a little tweaking.

magpi.cc/performingpumpkins

DECOR OF THE DAMNED

Interactive interiors that are sure to intimidate invited fiends

Remote-controlled Billy

david0429

Depending on the kind of party you're throwing, you might not want to greet your guests with this guy from the *Saw* movies as they walk through the door. Otherwise, it's a fun little thing for people to play with. Maybe you can build a Kermit on another one and have them duel like it's Pi Wars?

This is actually not a bad step up from some robot kits, and you could make the doll or find one on Etsy or something.

magpi.cc/billysaw

Possessed Portrait

Dominick Marino

While it may be a little conspicuous to suddenly have a giant portrait in your home, it would at least fit a Halloween aesthetic. Either way, whenever a friend goes to inspect it, they won't expect the picture to move and scream at them.

This works largely on the same hardware principle as a magic mirror, albeit without the reflections and kind messages. Finding an antique (looking) frame might be the way to go as well.

magpi.cc/possessedpic

Haunted Jack-in-the-Box

Sean Hodgins

This one can be hidden away in your home, just in sight of the party guests. Until... suddenly it starts and scares the living daylights out of someone. This uses an actual Jack-in-the-box, so make sure it's one you're happy to modify with some 3D printing first.

It also makes for a very effective scare, so perhaps gauge the level of scare-ability of your guests before subjecting them to it.

magpi.cc/hauntedjack

TRACK PEOPLE BEFORE GIVING A TERRIFYING OWL HOOT

DELIGHTFULLY DEVILISH DRINKS

These beastly beverages are sure to go down a (trick or) treat

Eyeball snot-tail

This mix of lime jelly, apple and/or pear juice, lemonade, lychees, cherries, and raisins creates a light, tasty citrusy mocktail that also happens to look like it's eyes in nondescript green goop.

magpi.cc/snottail

Mulder

Mike Cook

There are pumpkins outside, so how about something a little different inside? This interactive skull may look a little jovial, but it can also be quite scary. The eyes, jaw, and neck move so it can track people before giving a terrifying owl hoot.

You can also remote control it just in case you want to give a custom scare to your friends.

magpi.cc/38

Halloween punch

This spiced and spicy cherry juice punch has a bit of a bite, and not just from the tangy, fangy sweets.

magpi.cc/punch

Mulled apple juice

If you want to serve something a little more traditional and a little less gimmicky, this mulled apple juice is a great idea.

magpi.cc/mulledapple

COSTUMES ARE CREEPY

Dress to depress with these despicable duds

Halloween voice changer

RoscoP13

Want to be really anonymous at your Halloween get-together? Then having a Raspberry Pi mask your voice on the fly is the way to go. This tutorial not only shows you how to do the software and electronic hardware, but also how to install it into a mask so you can actually wear it.

With some modifications, you can give yourself a custom voice profile, whether you want to be Darth Vader or Bane.

magpi.cc/voicechanger

Time Circuit and Flux Capacitor

Carl Monk

You could try and cosplay as a time-travelling Irish sports car if you have enough cardboard and room to manoeuvre. However, it's a bit easier to cosplay as its inventor, with props from the car. These recreations use Raspberry Pi and seven-segment displays in a fun way that can be used for a lot of portable projects, and portable costumes.

The web page has a detailed tutorial on how to build the time circuit and such, although you'll need to find your own Doc Brown outfit.

magpi.cc/docbrown

Face-changing projection mask

Sean Hodgins

From the maker of the haunted Jack-in-the-box comes this incredible projection mask that uses a small DLP projector hooked up to a Raspberry Pi connected via a stick. The projector sits in the chin of the mask and uses projection mapping on specific videos so that they show up as they would on a screen. It's a very clever mix of hardware and software.

magpi.cc/projectionmask

Disco ball

Wolfie

No one will miss you at the party if you're walking around with many strips of LEDs attached to your person. This one can work on its own, but you can also hook it up to your house's light show if you want for some wonderful synchronised costuming.

There's lots of LEDs, chicken wire, foil, and some 3D-printed parts to make this, although you could always use the tech for something else.

magpi.cc/discoball

FIENDISH FINGER FEAST

A brutal buffet featuring both sweet and (un-) savoury delights

YOU COULD ALWAYS USE THE TECH FOR SOMETHING ELSE

Witches fingers

This finger food is quite literally fingers... or at least breadsticks with a garlicky tang to them.

magpi.cc/witchfingers

Dirt and worms

Gummy worms buried in crumbled biscuits and chocolate pudding? Spooky and delicious.

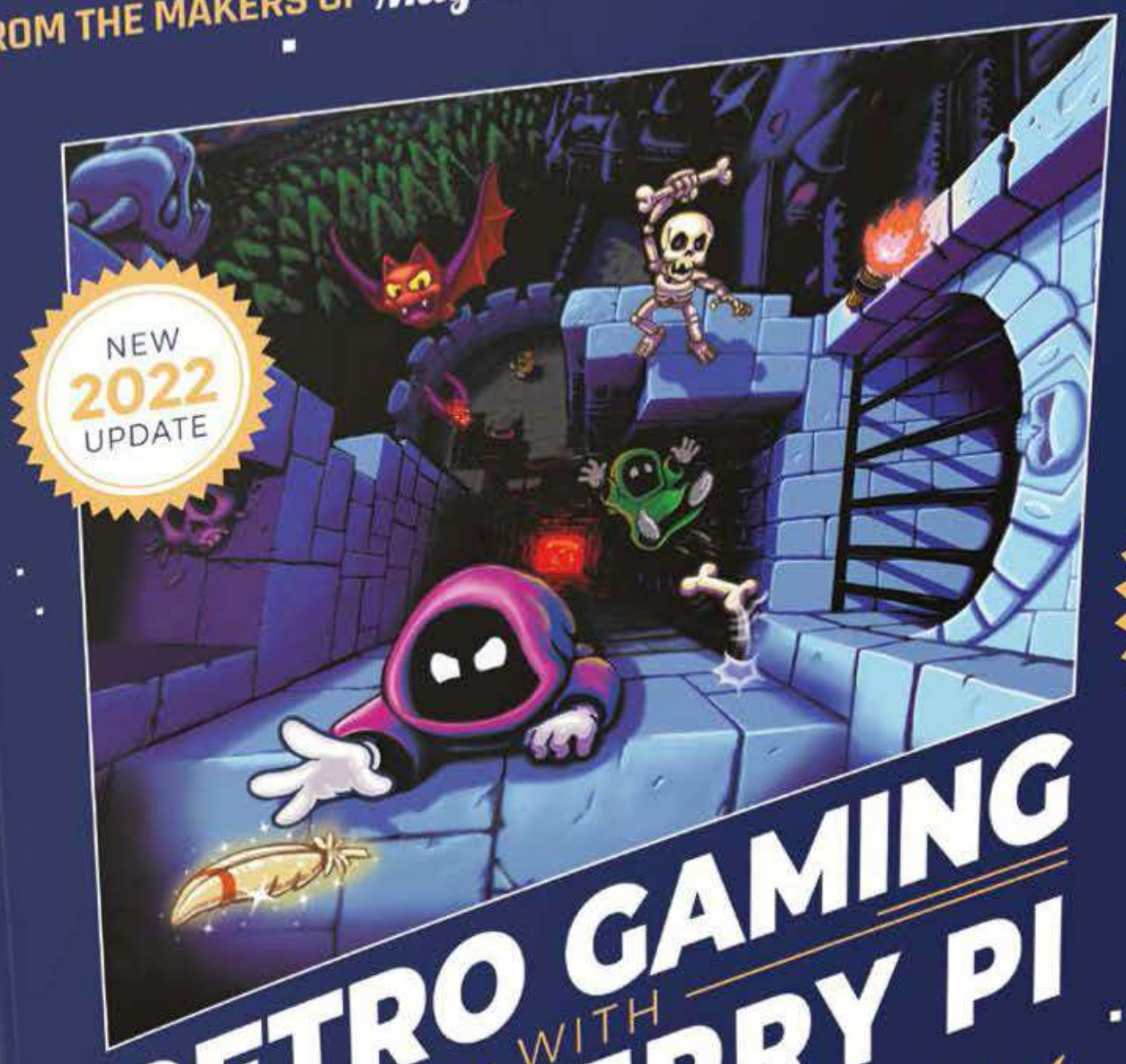
magpi.cc/dirtnworms

Sausage mummies

Cocktail sausages wrapped in spirals of puff pastry – add edible eyes or little dots of mustard to make them really stand out.

magpi.cc/sausagemummies

FROM THE MAKERS OF The MagPi THE OFFICIAL RASPBERRY PI MAGAZINE



NEW 2022 UPDATE

RETRO GAMING WITH RASPBERRY PI

2ND EDITION

164 PAGES OF VIDEO GAME PROJECTS

BUILD AN ARCADE MACHINE

PLAY & CODE GAMES!

RETRO GAMING WITH RASPBERRY PI

2ND EDITION

Retro Gaming with Raspberry Pi shows you how to set up a Raspberry Pi to play classic games. Build your own games console or full-size arcade cabinet, install emulation software and download classic arcade games with our step-by-step guides. Want to make games? Learn how to code your own with Python and Pygame Zero.

- ***Set up Raspberry Pi for retro gaming***
- *Emulate classic computers and consoles*
- ***Learn to code your own retro-style games***
- *Build a console, handheld, and full-size arcade machine*

BUY ONLINE: **magpi.cc/store**

EPD Pico Kit

SPECS

SCREEN:
2.66" e-ink display, 296 × 152 resolution, 125 DPI, two colour

DISPLAY DRIVER:
EXT3 board with a 90 degree header and 8 megabits of flash memory

SOFTWARE:
Pervasive Displays Library Suite software

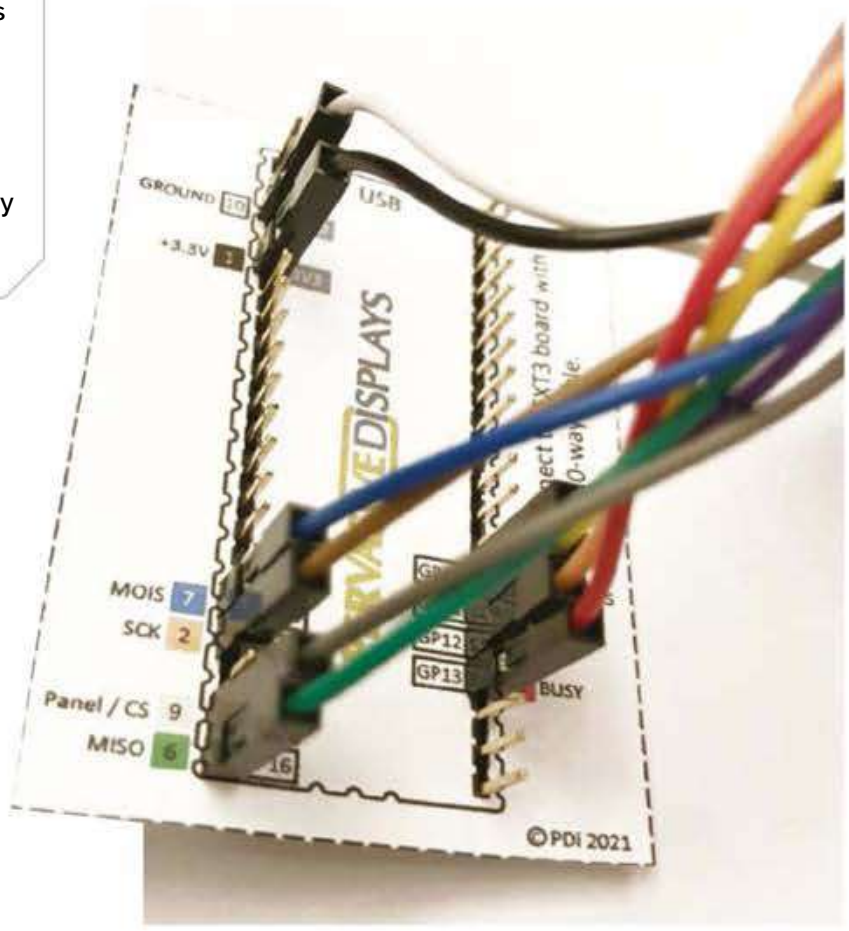
▶ Pervasive Displays ▶ **magpi.cc/epdk** ▶ £25 / $30

A cost-effective and easy way to get into using e-ink displays. By **Rob Zwetsloot**

This neat little kit is a great way to add a small e-ink display to a Raspberry Pi Pico – in fact one is included in the kit. It uses a specific extension board to connect to the screen and then to Pico using provided jumper wires – this does mean that it has a larger footprint than, say, the Badger 2040 from Pimoroni, which is used as a fun name badge.

This EPD is aimed at slightly different applications and users though. With full access to the rest of Pico, many more sensors and inputs can be connected to the system, allowing it to be used for temperature and air-con control – shown off quite nicely in the demo video and code.

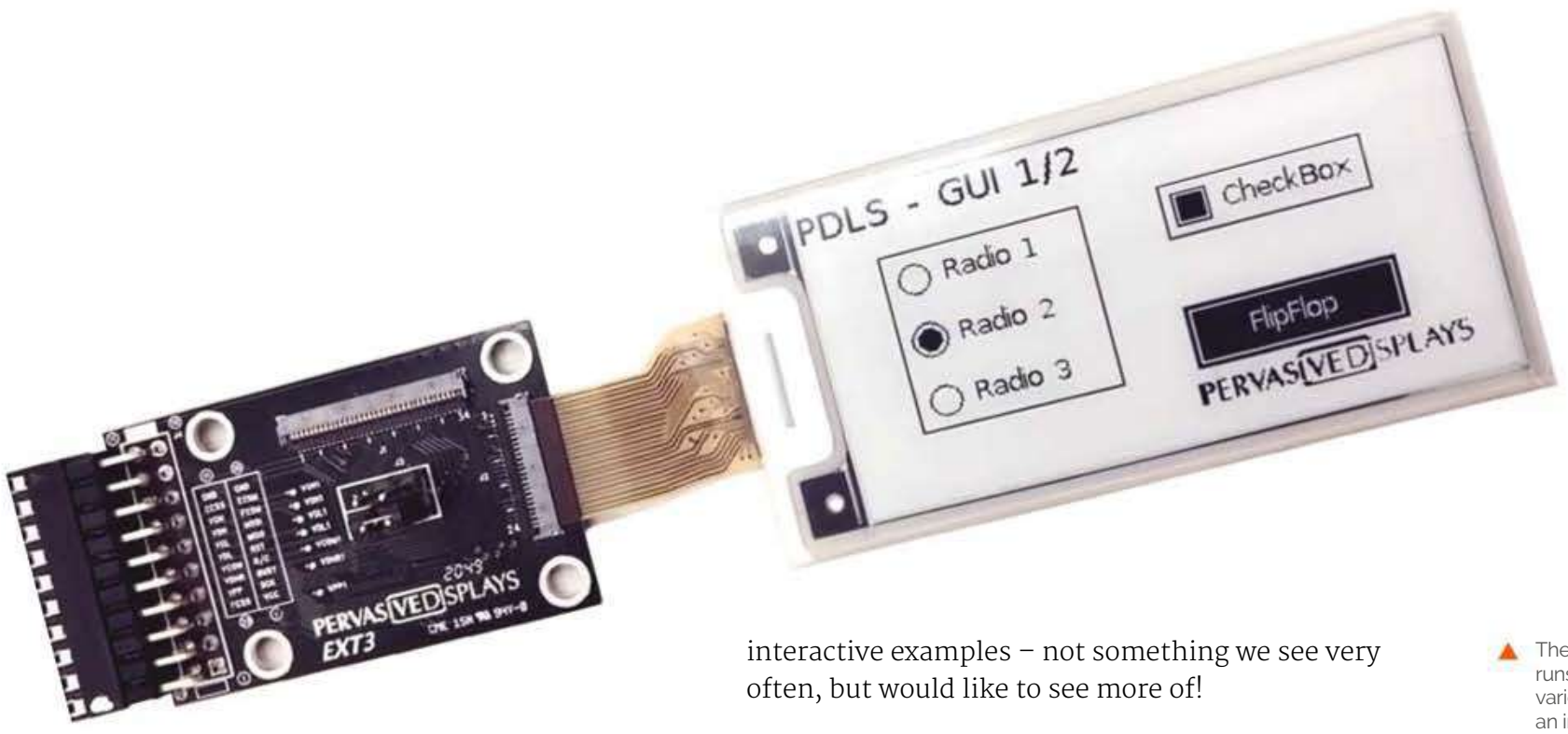
> With full access to the rest of Pico, many more sensors and inputs can be connected to the system



▶ The pin mapping guide is a handy way to make wiring up easier

Intense wiring

Putting it all together out of the box can take a little time. Roughly a dozen GPIO pins need to be connected to the right spots on Pico, albeit made easy thanks to a little pinout card that comes with the kit. These jumper wires connect to a header which, luckily, slots straight onto the extension board, with a ribbon cable connecting to the screen.



The example code runs you through various functions in an interesting way

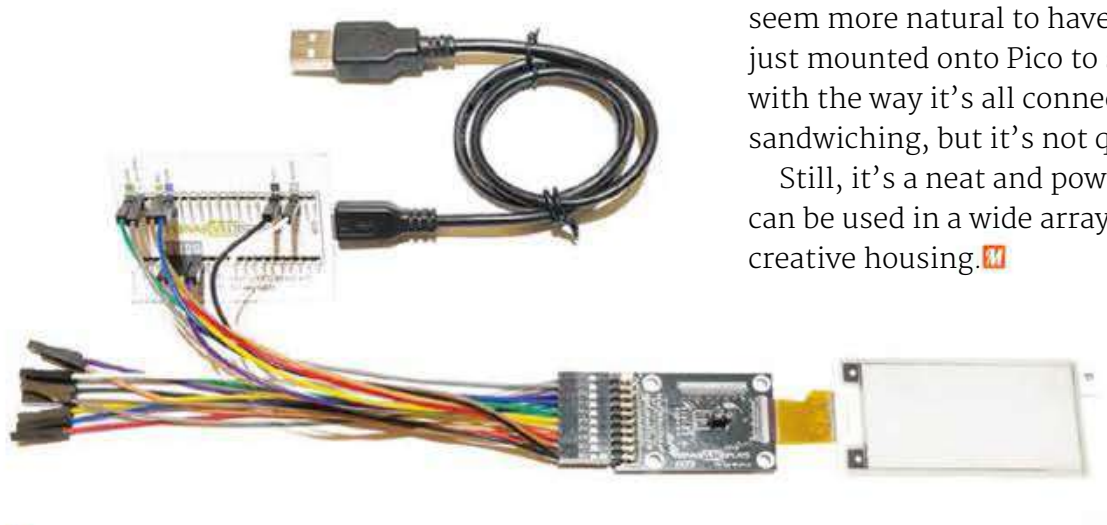
It's all powered off a Raspberry Pi Pico, so you won't need to connect extra power to the screen at all – a decent mobile battery should be sufficient to power the whole rig. Connect it to a PC and load the firmware and demo and, not only will you get a look at the aforementioned demo, but a rundown of the different functions in the code itself with interactive examples – not something we see very often, but would like to see more of!

In its place

The library for the screen is great and the code is fairly simple to understand, although it is in C as it has roots in Arduino, which might cause some issues for some people. C seems to run better on Pico though than MicroPython, so it was probably the right choice to get the most out of it.

The slightly clunky way it looks when put together basically does feel a bit of a shame though. Considering how small Pico is, it would seem more natural to have something like this just mounted onto Pico to save on space. Luckily, with the way it's all connected, you can do some sandwiching, but it's not quite the same.

Still, it's a neat and powerful little display that can be used in a wide array of settings with a bit of creative housing.



The whole set up is a little clunky, but it does allow access to all the features of Pico

Verdict

A very powerful and well thought out piece of kit, that feels like it could be a bit more condensed.

9/10

Autonomous Robotics Platform for Raspberry Pi Pico

SPECS

DIMENSIONS:
PCB length: 126 mm; PCB width: 80 mm; wheel diameter (with tyre): 67.5 mm

SENSORS:
Ultrasonic distance sensor HC-SR04 5 V; Kitronik line-following sensor board

MOTORS:
2 × TT geared motors

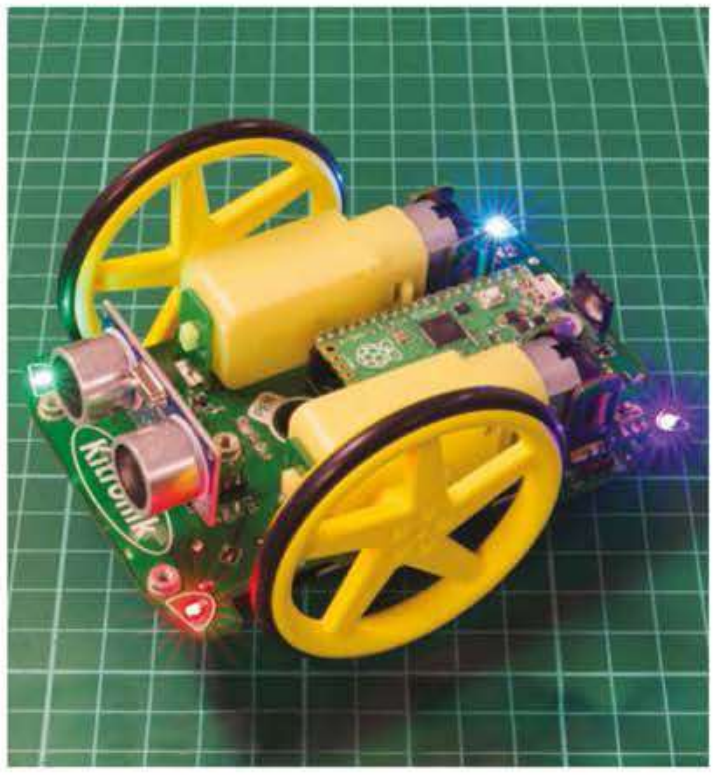
▶ Kitronik ▶ **magpi.cc/kitronikpico** ▶ £41 / $49

This fun robotics board uses Pico to great effect.

Lucy Hattersley takes it for a spin

Kitronik's Autonomous Robotics Platform caught our attention recently thanks to its usage of Raspberry Pi Pico, rather than the more common Raspberry Pi Model B or Zero models.

The kit contains a robotics platform chassis with two TT motors pre-mounted. Two large yellow wheels are attached to the side, along with an ultrasonic sensor on the top and a line-following sensor underneath. Finally, a Pico or Pico W with a GPIO header soldered in can be mounted in the middle of the two motors. Four AA batteries are slotted in underneath to provide power to the motors and Pico.



▶ Four LEDs light up the Kitronik Autonomous Robotics Platform

The result is a cute-looking robot that is easy to assemble, making it perfect for younger robotics makers. It's incredibly lightweight and moves around at a brisk pace.

All aboard

There are a few extras on the robot chassis worth mentioning. On all four corners sit ZIP LEDs that add bling (and can be useful for feedback). A hole in the middle of the board is used to hold a marker pen for turtle-like drawing. There is an on/off switch to cut the power and a button that responds to code (as opposed to the BOOTSEL button on Pico W). Finally, there's an on-board buzzer to make audio feedback.

We found it easy to set up, thanks to the included manual. At least to the point where the physical assembly was complete. Following the build, the manual skims over the API and mostly directs you to the Kitronik website (**kitronik.co.uk/pico-arp-motors**) for more detail on how to code and control the robot.

Clone the corresponding GitHub repo (**magpi.cc/kitpicogit**), and you'll discover code to go with all the tutorials and some great example programs. Along with tests for all the motors, sensors, button, and buzzer, there's code that runs the robot around in circles, line-following examples, pen-lifting examples, and a program that uses the sensors to control the lights.

Autonomous Robotics Platform is a good-looking robot that's easy to control

The GitHub page has documentation on the API, and the tutorials are comprehensive.

Using Pico instead of Raspberry Pi for the code has advantages and disadvantages. Even though Pico W is now available, you cannot remote-control the robot via a web or smartphone app (as you can with many other robots). Perhaps this functionality can be implemented down the line.

Ideal for a cost-minded learning environment

Pico runs code as soon as it's switched on, though, so the robot is functional in a code-and-drop way that makes it more reliable than Raspberry Pi running a full OS. And you're not faced with the usual SSH and wireless networking complication that troubles many a robot setup. You create code on your computer and drop it directly onto Pico to run.

We think this is a nice robot build that will be lots of fun. It packs a lot of features onto a board given its low cost (which is better value when you factor in running Pico, rather than a full-blown Raspberry Pi computer).

Autonomous Robotics Platform is ideal for a cost-minded learning environment. These robots are cheap to buy, easy to set up, sturdy, and fun to program.

The PCB board comes with two TT motors mounted; wheels, sensors, and Pico are attached to complete the build

Verdict

A great little robotics learning environment that is great value when you factor in the low cost of Pico. It's packed with features too.

9/10

Arducam 64MP Autofocus Camera Module

SPECS

SENSOR:
1/1.7" stacked CMOS image sensor, 0.8 µm pixel size

LENS:
f/1.8 aperture, 84˚ view angle, 8 cm–∞ focal range, motorised focusing

MAX RESOLUTION:
9152×6944 stills; 1080p 30 fps video

▶ The Pi Hut ▶ **magpi.cc/64mpcamera** ▶ £60 / $60

Zoom in to the fine detail with this super-hi-res camera. By **Phil King**

As its 'Hawk-eye' nickname implies, the 64MP Autofocus Camera Module can capture stills with unprecedented detail: up to 64 megapixels (9152×6944) on Raspberry Pi 4 or CM4 (other models are limited to 16MP).

▲ It's the same size as a standard Camera Module, although the lens is a little bigger; here it's shown without its case

> Rather than being fixed focus, the lens is motorised

That's right up there with the top camera phones. It's not as intuitive to use as a smartphone camera, though.

The 1/1.7-inch sensor is attached to a PCB of the same dimensions as a standard Raspberry Pi Camera Module. It comes in a neat little case with mounting holes and a tripod screw mount on the rear. A 15 cm CSI camera cable is supplied.

Rather than being fixed focus, the lens is motorised; you can hear it clicking in and out. This means the camera focus can be adjusted in the software – it uses a forked version of the standard libcamera library, installed along with custom drivers via a few Terminal commands.

▼ The supplied 15 cm CSI camera cable is used to connect to Raspberry Pi; you may want to get a longer cable or extender kit

Staying in focus

The autofocus (AF) option is a welcome feature, although it doesn't always work quite as expected. For instance, it'll typically focus on a busy background, so it's best to shoot subjects on a plain backdrop. Alternatively, you can use a simple utility to focus manually. Another smart option is continuous autofocus, which re-triggers AF whenever a change is detected in the scene. There's also a digital zoom (up to 10×) option that enables you to move the preview around the live scene and zoom in and out.

Indoor shots under artificial lighting came out rather dark, but this can be corrected with parameter tweaks such as extra exposure. 64MP stills also suffered from tiny horizontal banding streaks in places and tended to be a little soft-focus, due to lens diffraction, but this can be fixed by sharpening in an image editor.

Verdict

With a single lens, it's not as versatile as the HQ Camera, but the motorised focusing is neat and it can shoot stills at an incredibly high resolution.

8/10

HackSpace

TECHNOLOGY IN YOUR HANDS

THE MAGAZINE FOR THE MODERN MAKER

SUBSCRIBE AND SAVE UP TO **35%** on the cover price

ISSUE **#58**

OUT NOW

hsmag.cc

Available on the App Store

GET IT ON Google Play

10 Amazing:

Gaming Accessories

Make your Raspberry Pi a lean, mean, gaming machine

We know lots of folks that use a Raspberry Pi to play retro games and homebrew. It's an easier way to hook it up to your TV after all. While you can just bodge together a gaming Raspberry Pi, there are many things that will help make it just that little bit better.

8BitDo Pro 2

Retro controller

With a classic retro design updated with modern conveniences (such as Bluetooth), the Pro 2 controller is one of the best ways to experience old games. It works on Raspberry Pi, PC, Switch, and more!

8bitdo.com | £42 / $50

NESPi 4 Case

Retro camouflage

This cheeky case will make your Raspberry Pi seem right at home among other consoles – there's also a removable cartridge for extra storage.

magpi.cc/nespi4 | £28 / $34

GPi Case

Handheld gaming

This familiar case is a genuinely great way to have a bit of retro gaming in your (big) pocket. It even has extra buttons so you can play more modern games as well.

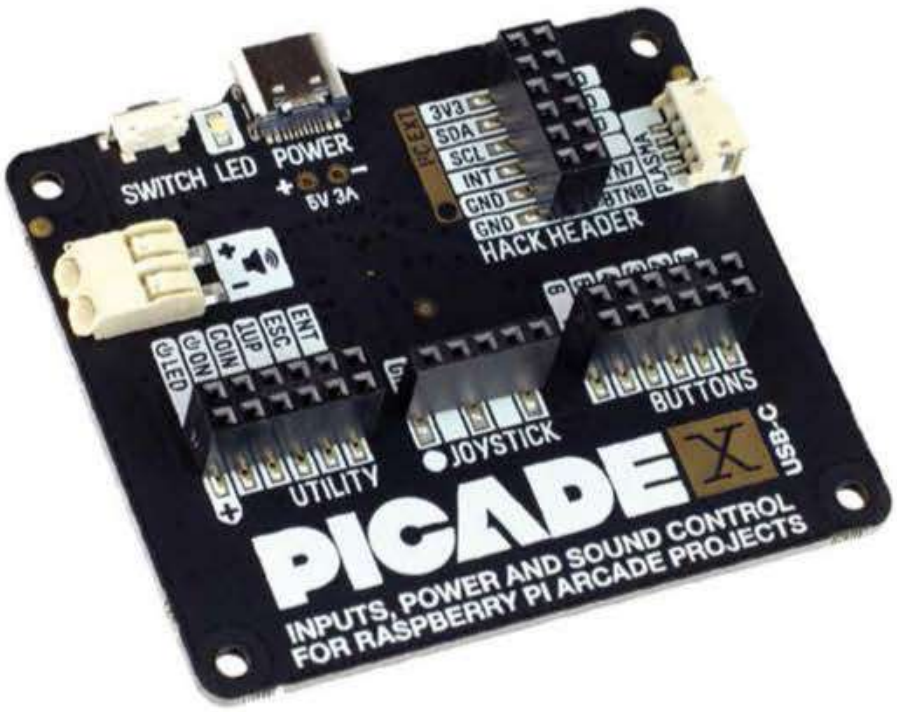
magpi.cc/gpi | £60 / $72

Massive arcade button

For big hits

This huge button is 100 mm across, and comes with an LED as well. It's perfect for very specific arcade and gaming builds, especially if you need to smack a big button very fast.

magpi.cc/massivebutton | £8 / $10



Picade X HAT USB-C

Ultimate arcade board

Created for the amazing Picade, the X HAT is the absolute perfect accessory for turning a Raspberry Pi into an arcade machine. Just build the cabinet, add the buttons, and you're good to go.

magpi.cc/xhat | £16 / $19

8BitDo Arcade Stick

Multi-format stick

Want to get your Street Fighter on like you're at an arcade machine? You can't go far wrong with this arcade stick. It can be wired and wireless, and has many switches to let it work in any way you wish.

8bitdo.com | £78 / $90

RGBerry SMA

Premium RGB HAT

This HAT is designed to connect a Raspberry Pi to an old CRT TV or video monitor much better than the standard analogue video out on Raspberry Pi. See those beautiful scanlines.

magpi.cc/rgberry | £30 / $36

Sanwa 8-Way Joystick

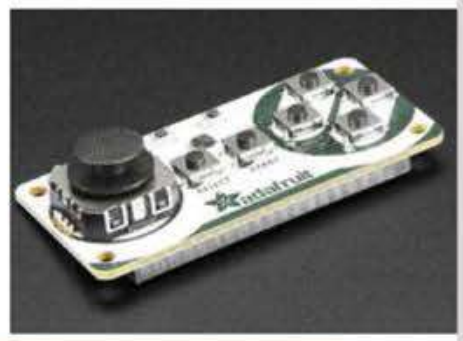
Premium movement

Sanwa parts are the gold standard for arcade controllers, and most arcade sticks you get can be easily converted to use one of these sticks. You can even change the gates on them – the corners of the stick.

magpi.cc/sanwajoy | £27 / $33

Joy Bonnet

Tiny controller HAT



This funky add-on for Raspberry Pi Zero turns it into a controller that is its own console. It's a bit fiddly, but it's also an excellent party piece, and can even fit in your pocket.

magpi.cc/joybonnet | £15 / $18

Keybow 2040

Extra hot keys

Need more keys for your gaming? Look no further than the Keybow 2040; 16 mechanical keys based on the same chip as Raspberry Pi Pico. You can program it with custom commands for gaming macros, or stream hot keys.

magpi.cc/keybow | £50 / $60

Learn electronics with Raspberry Pi

Resources to help you get started with electronics on Raspberry Pi. By **Phil King**

Raspberry Pi Projects

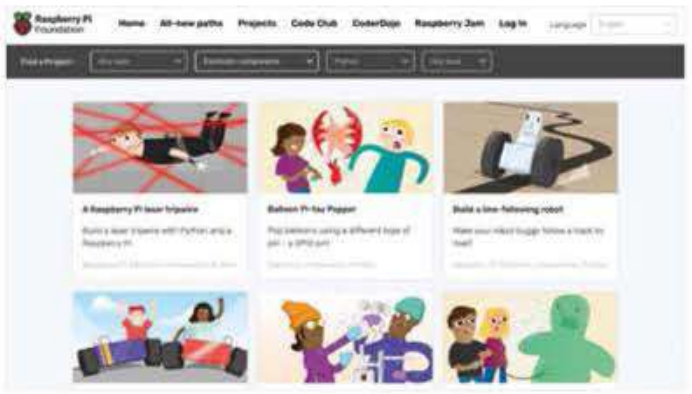
AUTHOR
Raspberry Pi Foundation

Price:
Free

magpi.cc/electronicprojects

While a Raspberry Pi is powerful enough to be used as a desktop PC, what makes it special is its 40-pin GPIO (general-purpose input/output) header. As well as being used for connecting add-on HAT boards, the GPIO makes it possible to hook up all sorts of standard electronic components, usually on a breadboard.

The Raspberry Pi Foundation's learning resources website is a great place to get started with building your own circuits and writing programs on Raspberry Pi to control them. It features a wide range of electronics projects, filterable by topic, hardware (including Pico microcontroller), software, and difficulty level.



For electronics newbies, we recommend the Physical Computing with Python tutorial. There are also multi-stage paths introducing physical computing with Scratch (**magpi.cc/scratchpath**) and digital making with Pico (**magpi.cc/picopath**). Other fun projects to make include a laser tripwire, wire loop game, burping Jelly Baby, and spinning flower wheel.

Cool kits

Grab a bunch of useful components in a handy kit

MONKMAKES PROJECT BOX 1

Make the ten projects detailed on instruction cards in this kit from Simon Monk which includes everything you need to make them: LEDs, a thermistor, phototransistor, switches, and more.

▶ **magpi.cc/projectbox1**

CAMJAM EDUKIT #2

This inexpensive beginner's kit introduces you to sensors: a PIR (movement), LDR (light), and DS18B20 probe (temperature). Detailed worksheets take you step-by-step through the projects.

▶ **magpi.cc/camjamkit2**

WAVESHARE SENSORS KIT

As used in our Sensory World tutorial series (issues 111 to 114), this kit comprises 13 sensors, including gas, moisture, and tilt. You'll need an ADC (analogue-to-digital converter) chip to read analogue signals from some of them.

▶ **magpi.cc/wavesensors**

CamJam EduKit #1

AUTHOR

CamJam / The Pi Hut

Price:
£6 / $6

magpi.cc/camjamkit

Electronics kits can provide a great way to get started, and this is one created by CamJam (Cambridge Raspberry Jam) is ideal for complete newcomers. Compatible with any Raspberry Pi computer, it contains a 400-point breadboard, three LEDs (red, yellow, and green), a push-button, piezo buzzer, resistors, and jumper wires.

More importantly, there's a set of excellent downloadable worksheets that feature step-by-step instructions, with thorough explanations and Python code – which makes use of the GPIO Zero library. There

are even Scratch scripts available if you prefer that approach. Projects include traffic lights and some simple games.

Once you've mastered the basics with this kit, you may well want to move on to the CamJam EduKit #2 to play with some sensors (see 'Cool kits' boxout).

Simple Electronics with GPIO Zero

AUTHOR

Phil King

Price:
Free PDF

magpi.cc/bookgpiozero

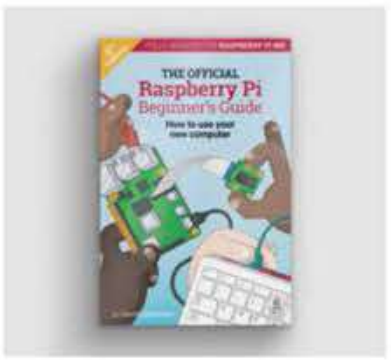
You can download this free PDF of this handy Essentials guide from *The MagPi* website. It takes you through the basics of connecting electronic components to Raspberry Pi and controlling them, before moving on to building some more advanced projects.

In the first few chapters, you'll learn how to connect common electronic components, such as LEDs and push-buttons, to Raspberry Pi's GPIO pins and control them with Python programs. The GPIO Zero Python library (**magpi.cc/gpiozero**) makes it a whole lot easier by eliminating a lot of boilerplate setup code for the GPIO pins.

While learning about electronics and coding, you'll make fun projects including an internet radio, motion-sensing alarm, LED thermometer, and basic robot. Clear wiring diagrams are provided for each tutorial, along with all the code needed.

Books to read

Recommended reading for learning electronics



THE OFFICIAL RASPBERRY PI BEGINNER'S GUIDE

As well as providing a thorough introduction to using Raspberry Pi itself, the official guide covers numerous electronics projects (including using the Sense HAT), based on both Python and Scratch.

magpi.cc/bgbook

RASPBERRY PI COOKBOOK 3RD EDITION

Simon Monk's 608-page tome features several chapters dedicated to connecting electronics such as LEDs, push-buttons, and sensors. An updated 4th Edition is due out in December of this year.

magpi.cc/rpicookbook

GET STARTED WITH MICROPYTHON ON RASPBERRY PI PICO

This official guide for Pico covers a range of electronics projects for the microcontroller, as well as using the I2C and SPI protocols and controlling NeoPixel strips with Pico's Programmable IO.

magpi.cc/picobook

Sam Alder

The in-house illustrator and animator for Raspberry Pi sets the look for a lot of what you see

> Name Sam Alder | > Occupation Illustrator and animator
> Community role Illustrator | > URL samalder.co.uk

You may have noticed that we sometimes have excellent illustrated covers – such as issue 120's robot cover, or even cool graphics throughout a feature. If you've ever thought they look similar to some official Raspberry Pi art, you'd be right, as they're also by the excellent Sam Alder, illustrator and animator for Raspberry Pi. "Or as I like to describe it: I draw things and sometimes make them move – whether they like it or not," clarifies Sam.

What is your history with drawing?

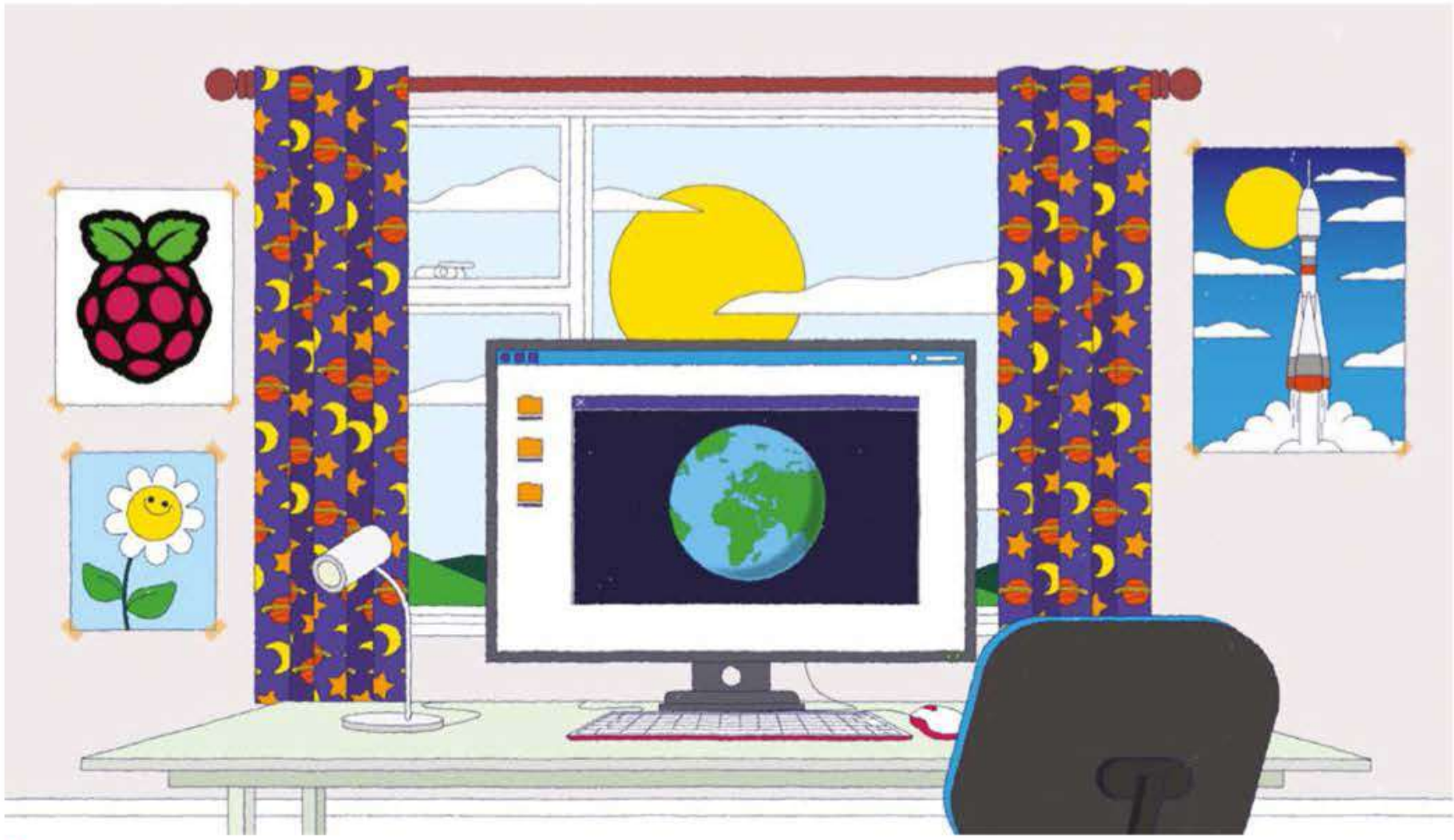
Growing up, I was always a massive doodler, first copying my favourite cartoons like *Ren and Stimpy*, *Earthworm Jim*, and *Rocko's Modern Life*, and later drawing my own comics and weird artwork. However, it was never something I studied properly at school or college. It wasn't until I got some seriously bad A level results, and was at a loose end as to what to do with my life, that my Mum and Step-Dad encouraged me to sign up to an art foundation course at my local college. It was a decision that totally changed my life.

From there, I went on to study illustration with animation at Manchester School of Art and, after graduating, went on to work at MTV before starting my own animation company, with close friend and collaborator Scott Lockhart directing music videos and TV adverts.

The book covers use a lot of illustrated elements created by Sam

How did you learn about Raspberry Pi?

I first heard about Raspberry Pi back in 2013 whilst sharing an office space in Manchester with Raspberry Pi veteran, Ben Nuttall. He would enthusiastically try to explain what it was all about, which didn't make a huge amount of sense at the time, but it piqued



▲ The animation from Sam that celebrated ten years of Raspberry Pi: **magpi.cc/10yearvid**

my interest, so I got in touch with my now boss Liz Upton to ask if we could make an animation to explain the story of Raspberry Pi. She kindly agreed and put a huge amount of trust in our tiny team. Our friendship was born, and we went on to make a further four animations over the next 18 months.

> "We went on to make a further four animations over the next 18 months"

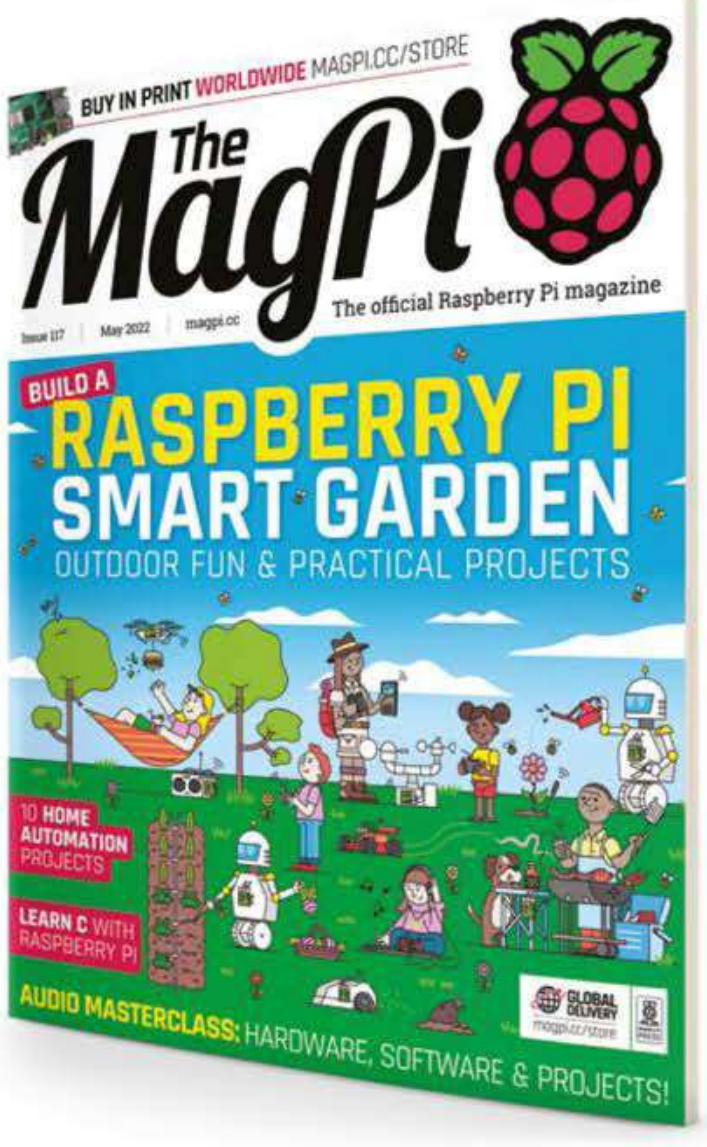
How did you get your job at Raspberry Pi?

In 2014 I decided to move on from our animation company and go at it alone. It was around that time that I was invited to join the, then relatively small, team at Raspberry Pi, which was an opportunity I couldn't turn down.

Since then, I've made lots more animations and illustrated tons of online resources, books, and magazines, including this fine publication!

What other hobbies do you have?

I love making music with my brother and have played guitar, amongst a few other instruments, for nearly 25 years. I was lucky enough to spend my twenties recording, touring, and performing with various bands, so music is a massive part of my life. Other than that: making pies, watching and playing cricket (incredibly badly), and exploring the Peak District, where I live with my partner Jane, are how I like to spend my time. Gosh, my hell-raising days are certainly behind me aren't they!



▲ Our Smart Garden cover was created by Sam, based on what we wanted to put in the feature

MagPi Monday

Amazing projects direct from our Twitter!

Every Monday we ask the question: have you made something with a Raspberry Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made.

Here's a selection of some of the awesome things we got sent this month – and remember to follow along at the hashtag #MagPiMonday!

01. Prototyping with cardboard boxes is a time-honoured tradition

02. RFID music players are a thing we've seen in the past, and they're always pretty great

03. These infinity mirror goggles on this hat are ingenious, really making them stand out from the crowd

04. More folks are making clean train departure boards, and this one on an e-ink display is great

05. We hope the price of that USB extension didn't... sting. We'll see ourselves out

06. We do love a good Node-RED project – using it to light up LEDs is a Dr Lucy Rogers classic, and it's great to see it done with NeoPixels

07. A smart way to maintain the life of this mini OLED display

08. BirdNET-Pi is used to identify different bird sounds, so this is a very creative housing for it

09. Instead of hunting for ghosts, Kevin is hunting for the optimum WiFi channel, with a Pico W!

10. We could absolutely do with this to-do list some weeks

04

James Linnegar
@MrLinnegar

Replying to @TheMagPi

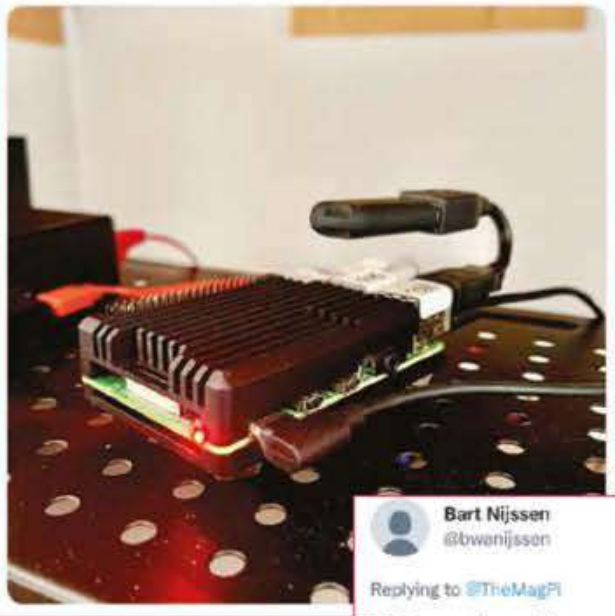
A custom train departure board for my two closest stations :)

05

Charlie O'Hara
@whalecomer

Replying to @TheMagPi and @Raspberry_Pi

Made my Raspberry Pu home automation server look "slightly" too aggressive and scorpion-like with a bendy USB extension.



06

HomeMadeGarbage
@HOmeMadeGarbage

Replying to @TheMagPi

I enjoyed LED lighting using Raspberry Pi Zero 2 W with node-RED.
#MagPiMonday

07

thinklearndo
@thinklearndo1

Replying to @TheMagPi

Updating my garage door state indicator to use inverting images so the OLED display doesn't get screen burn in. There is a @Raspberry_Pi in there :)
project files: github.com/thinklearndo/g...

08

Bart Nijssen
@bwnijssen

Replying to @TheMagPi

This weekend I created an outdoor bird house for my new #birdNET-PI

09

Kevin McAleer
@kevsmac

Replying to @TheMagPi

Happy hot #MagPiMonday !
I made a Wifi Hotspot scanner using a Raspberry Pi Pico W, (it's also inside this cool 3d printed Ghostbusters case and yes, the arms move based on Signal strength). Download the code here:
github.com/kevinmcaleer/g...
#ghostbusters #raspberrypi #micropython

10

AxWax (@axwax@fosstodon.org)
@AxWax

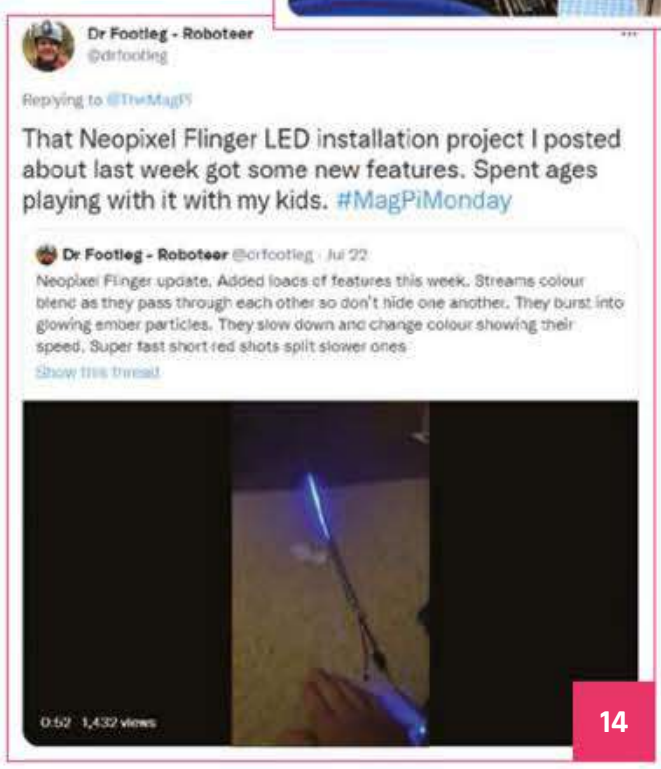
Replying to @TheMagPi

I've had a play with my new @Raspberry_Pi Pico and @pimoroni's Pico Inky Pack.

I managed to grab server time, custom images and my to-do list via @trello's API from my server, using buttons to switch.

Video of it in action: twitter.com/AxWax/status/1...

#MagPiMonday #MicroPython



11. Having a slide show display of different JWST and NASA photos is a great way to improve your desk we think

12. Martin was also able to see the photo elsewhere a bit more clearly, but this is great for glancing at

13. If you're in the UK at the moment, making sure your paddling pool is nice and cool is a must

14. This Finger LED thing seems to be a game in the making – check out Dr Footleg's Twitter feed for the videos, they're very cool

Crowdfund **this!**

Raspberry Pi projects you can crowdfund this month

CodeRover

This 'sustainable and programmable robot' is a great robotics platform that allows you to build, modify, and program as you wish. It uses a 'core' that you plug into a Raspberry Pi or a microcontroller and hook up via jumper cables to the rest of the system. Once that's done you can program it with the block-based Code:Bit platform built for the robot.

▶ **magpi.cc/CodeRover**

EncroPi

A neat USB data logger and RTC that allows you to encrypt data on it – and it uses an RP2040 that powers Raspberry Pi Pico. It even has a little LCD screen on it so you can see what time is being kept on the RTC.

▶ **magpi.cc/EncroPi**

Your Letters

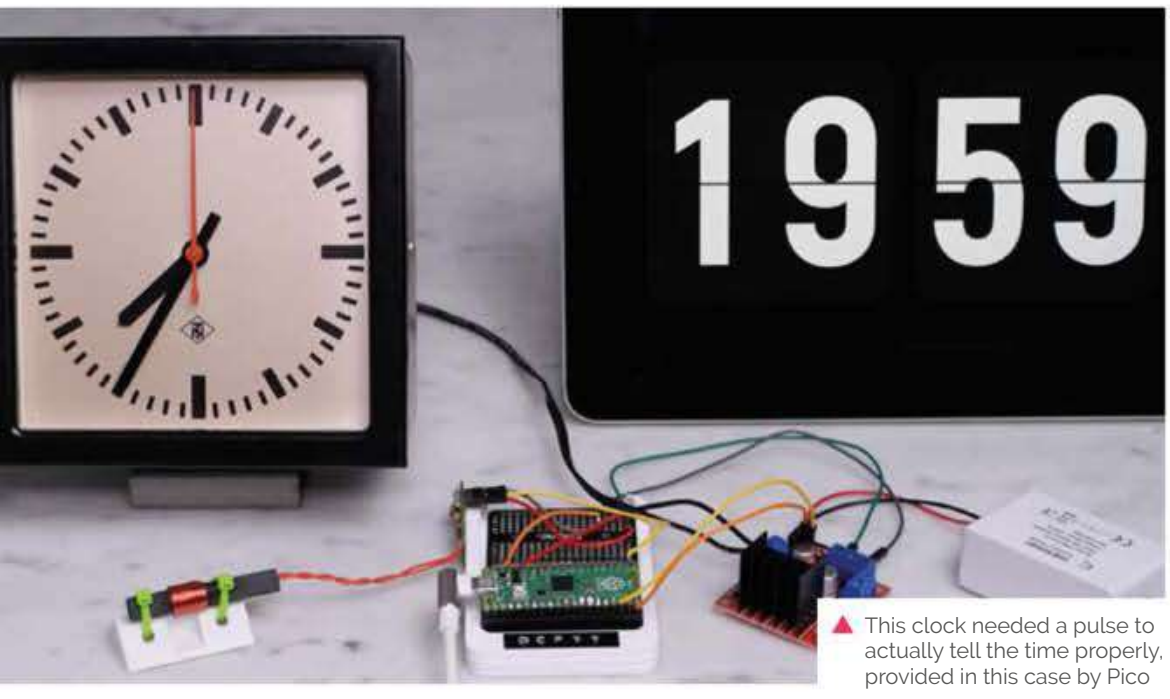
More mother clocks

[In reply to a tweet about the Pico railway clock from issue 120 – Ed] My old uni had something like this through our building. Someone cut a wire somewhere and killed the whole system until it was traced.

I thought it received a 1Hz 'clock' pulse (you see what I did there!). The system could send multiple pulses to adjust for summer time hours.

Steve via Twitter

That sounds like a bad way to find out about that system – we had other people tell us of similar systems, such as factory clocks. Again, it's wild that such seemingly ubiquitous tech has been lost to time, especially as it seemed to work pretty well – despite the hazard of being cut.



▲ This clock needed a pulse to actually tell the time properly, provided in this case by Pico

Embedded events

I saw that there was a Raspberry Pi presence at Embedded World that I completely missed. Are there any other events you'll be at in the near future?

Amy via email

Yes, various teams will be showing up at events around the world as places open up more – the next one will be SiliCon with Adam Savage in California, which will be starting the weekend after this issue is released (27 and 28 August). Features Ed Rob will be there too, sporting a fancy green Luigi's Mansion cosplay, so look out for him!

▲ Keep an eye on Raspberry Pi's social media for info about when the teams will be at different events

Contact us!

- Twitter **@TheMagPi**
- Facebook **magpi.cc/facebook**
- Email **magpi@raspberrypi.com**
- Online **forums.raspberrypi.com**

▲ Wayne Chan's notifier was used to make sure people knew how many others were in the office

WiFi extension

Regarding the Entry/Exit Notifier [issue 117 – Ed] – Mr Chan can extend his WiFi dongle range by attaching the dongle to a USB cable and placing the dongle through a hole in a soda, soup, or Pringles can about an inch from the bottom of the can. Then aim the can in the direction of the nearest WiFi access point. I have used this cantenna successfully between my and my neighbour's house when I temporarily shared my internet connection with him, as his connection was out of service and he was under a deadline for his business. His house is about 100 yards from mine.

Buz via email

Ah yes, we sometimes forget about some of the more interesting range extension hacks for WiFi. This one can be useful for anyone – although make sure that you have permission, like Buz's neighbour, for anything like this.

Wireframe

Join us as we lift the lid
on video games

Visit **wfmag.cc** to learn more

WIN

CROWPI L BASIC KIT

An amazing laptop body for your Raspberry Pi, CrowPi L is the lighter version of CrowPi2, yet is still just as useful. It has a webcam above the screen, and provides direct access to specific ports on Raspberry Pi. We have one kit to give away.

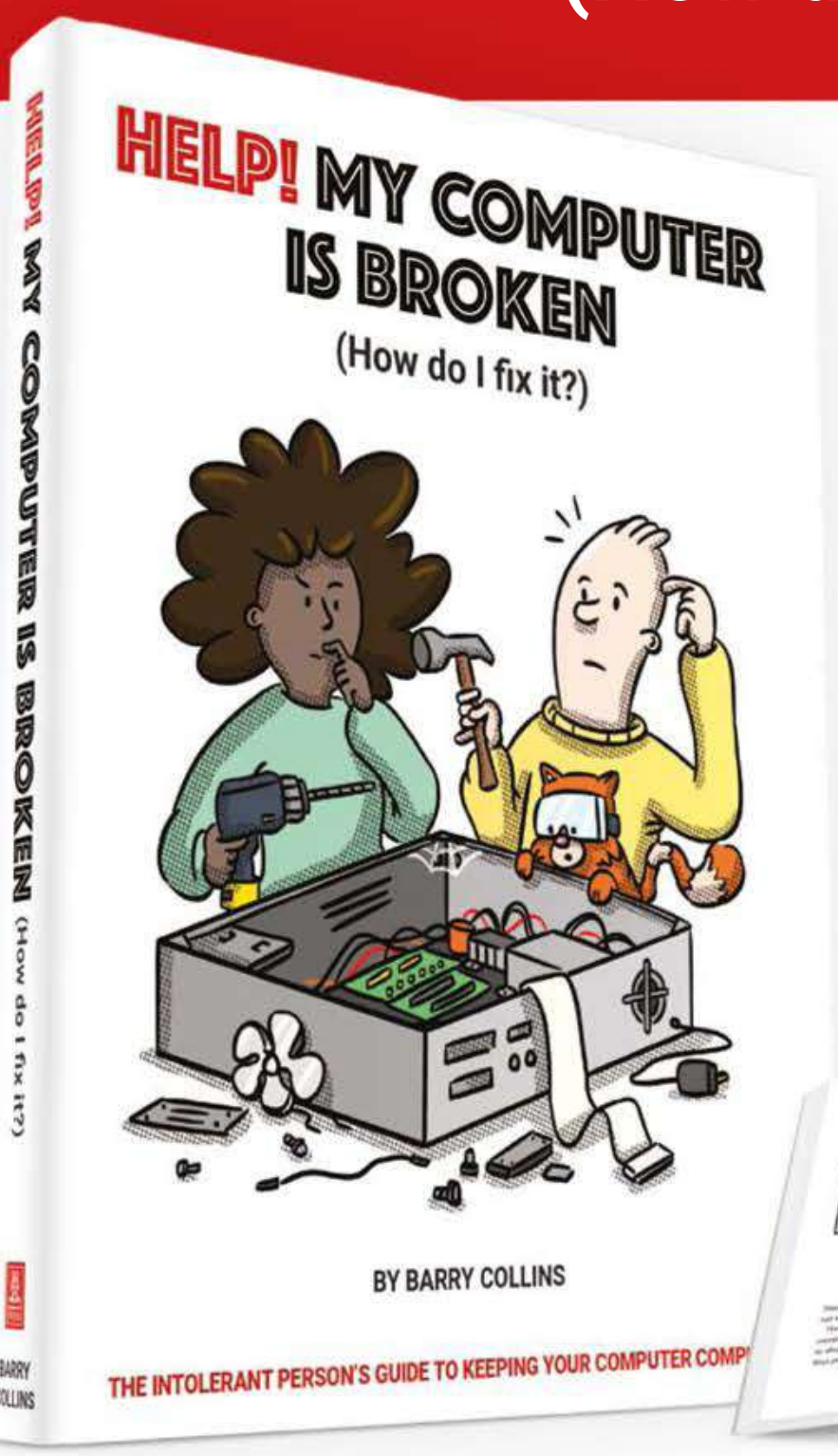
Head here to enter: magpi.cc/win | Learn more: magpi.cc/crowpil

Terms & Conditions

Competition opens on **24 August 2022** and closes on **29 September 2022**. Prize is offered to participants worldwide aged 13 or over, except employees of Raspberry Pi Ltd, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from The MagPi magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram, Facebook, Twitter or any other companies used to promote the service.

HELP! MY COMPUTER IS BROKEN

(How do I fix it?)



Help! My Computer Is Broken takes the most common computer problems and tells you how to fix them. It's as simple as that! If you've ever wondered why your laptop won't turn on, you can't get a WiFi connection, your printer isn't printing, or why everything is so slow – well, this is your book…

BUY ONLINE: magpi.cc/helpbook

THE MAGPI **#122**
ON SALE **29 SEP**

Plus!

Build a Magic Mirror

Petoi Bittle: the AI dog

Deep dive with a LEGO RC submarine

DON'T MISS OUT!
magpi.cc/subscribe

TWITTER	@TheMagPi
FACEBOOK	fb.com/MagPiMagazine
EMAIL	magpi@raspberrypi.com

EDITORIAL

Editor
Lucy Hattersley
lucy@raspberrypi.com

Features Editor
Rob Zwetsloot
rob@raspberrypi.com

Sub Editor
Nicola King

ADVERTISING

Charlotte Milligan
charlotte.milligan@raspberrypi.com
+44 (0)7725 368887

DESIGN

criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Olivia Mitchell, Ty Logan, Sam Ribbits, Lee Robinson

Illustrator
Sam Alder

CONTRIBUTORS

Alasdair Allan, David Crookes, PJ Evans, Ben Everard, Rosemary Hattersley, Nicola King, Phil King, Sean McManus, Stephen Smith

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.com

Director of Communications
Liz Upton

CEO
Eben Upton

DISTRIBUTION

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS

Unit 6 The Enterprise Centre
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE
+44 (0)1293 312193
magpi.cc/subscribe
magpi@subscriptionhelpline.co.uk

recycle

CC BY NC SA

This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

The MagPi magazine is published by Raspberry Pi Ltd, Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products, or services referred to or advertised in the magazine. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).

Raspberry Pi PRESS

ISSN: 2051-9982.

Wearing Raspberry Pi

Did you notice that Features Ed **Rob Zwetsloot** likes to write about wearables and cosplay? There's a good reason for that

For a very long time, I've been into dressing up. I vividly remember being 17 and going to every charity shop in town looking for a specific type of striped trouser I could wear for an Ace Ventura outfit. It wasn't even for a specific or important event. It was just fun.

Fast forward several years, and suddenly I'm attending comic cons and anime conventions, and seeing a lot of people having fun dress up as their favourite fictional characters, so I join in. The more I participated and learned about cosplay, the more I noticed the creativity and craft behind it – and the impressive electronics people were installing.

As the internet, YouTube tutorials, and making in general, has blown up over the last decade, so too has the complexity of costumes and props. A friend of mine has 3D-printed a Buster Sword from Final Fantasy VII, complete with glowing Materia orbs from the game, something you just wouldn't see in the 2000s.

Getting stuck in

As a maker myself, and writing for *The MagPi*, I have definitely been wanting to make some great and grand things for cosplay with Raspberry Pi. My first foray was some flashing NeoPixels on Raspberry Pi Zero for my friend Freya and her Sans (from video game Undertale) cosplay. You can find it at **magpi.cc/neopixeleyes**, and it ended up working pretty well.

> I'm basically upgrading a Luigi's Mansion costume to be a bit more of a spectacle

Programming NeoPixels has come a long way since then, and is far easier on Raspberry Pi Pico. However, I did learn some limitations. Buttons in Python scripts need to be very carefully managed to make sure each press is registered – something GPIO Zero is mostly successful with, but can be easier with Pygame due to the way it threads stuff. Also, soldering NeoPixels correctly can be very tricky if you don't have much soldering experience. Despite having done this and helped other pals work on their own cosplay electronics, I still haven't got around to it myself, despite my previously mentioned grand plans.

Into action

As you read this, I'll be in my way to the Bay Area in California for SiliCon with Adam Savage. Raspberry Pi has a booth there and, as well as showing people Raspberry Pi projects on the stand, I'll be walking around in a Pico-powered cosplay. I'm basically upgrading a Luigi's Mansion costume to be a bit more of a spectacle, and I'm really looking forward to showing it off to you in the next issue, including a tutorial on how to replicate it!

It's really cool how making is becoming so much more easier and accessible to include in your other hobbies.

AUTHOR

Rob Zwetsloot

Rob had to 3D-print a hoover this month for important work-related reasons. He refuses to provide more context.

magpi.cc

HiPi.io

HIGHPI PRO

The new case from the HiPi.io team

- Rapid tool-free assembly and disassembly
- Large internal volume for HATs
- Compatible with Pi 2/3/4
- Multiple lid options
- Passive & Active Cooling options
- Secure microSD card cover
- VESA mount support
- Molding-configurable output ports customizable for volume orders
- Printed logo for your branding

Available at these great Pi stores:

PiShop.us PiShop.ca PiShop.mx ThePiHut Pi-Shop.ch Welectron.

Contact your favorite Pi store if it's not listed here

HiPi.io

PiKVM

Manage your servers or workstations remotely

A **cost-effective** solution for data-centers, IT departments or remote machines!

PiKVM HAT
for DIY and custom projects

Pre-Assembled version

- Real-time clock with rechargeable super capacitor
- OLED Display
- Bootable virtual CD-ROM & flash drive
- Serial console
- Open-source API & integration
- Open-source software

Available at the main Raspberry Pi resellers

core ELECTRONICS · PiShop.ca · RPishop.cz Váš dodavatel Raspberry Pi · Welectron. · elektorstore · PIMORONI

Pi-Shop.ch · thePiHut · PiShop.us · CloudFree · PiShop.mx

Reseller suggestions and inquiries:
wholesale@hipi.io